A Collection of the AIAA MODELING and SIMULATION TECHNOLOGIES CONFERENCE Technical Papers

Portland, Oregon
9–11 August 1999



For permission to copy or republish, contact:
American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive
Suite 500
Reston, VA 20191–4344

A Collection of Technical Papers

AIAA MODELING and SIMULATION TECHNOLOGIES CONFERENCE

Portland, Oregon
9–11 August 1999

Copyright © 1999 by
American Institute of Aeronautics and Astronautics

ISBN
1-56347-301-1

All rights reserved. No part of this volume may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without permission from the publisher.

Printed in the U.S.A.

A Collection of the AIAA Modeling and Simulation Technologies Conference 9–11 August 1999 Portland, Oregon

Table of Contents

Paper No.	Title and Author	Page
Session 13-MST-1	**Avionics, Weapons, and Engagement Modeling and Simulation**	
AIAA-99-4021	Rapid Prototyping Tools Applicable to Avionics Requirements Development and Real Time Manned Simulation C. Bendall, W. Bezdek, D. Green, and T. Trapp	1
AIAA-99-4022	An Avionic System Modeling Framework R. Bluff	12
AIAA-99-4023	Traffic Information Display Enhancing Pilot Situation Awareness: PART I K. Funabiki, S. Tenoort, and F. Schick	22
AIAA-99-4024	Real Avionics in Flight Simulators for Military Aircraft Y. Gindin and A. Peleg	32
AIAA-99-4025	Swedish Experiences from Air Defense Models E. Berglund and B. Hansson	41
AIAA-99-4026	Missile Model Fidelity and Integration for Air Combat Simulators P. Zipfel and T. O'Grady	49
Session 14-MST-2	**Human Factors and Cueing Systems**	
AIAA-99-4027	Tools for Including Realistic Representations of Operator Performance in DoD Constructive Simulations E. Martin, B. Brett, and D. Hoagland	55
AIAA-99-4028	Are Criteria for Motion Cueing and Time Delays Possible? R. Hosman	68
AIAA-99-4029	Analytical Evaluation of a Simple Partial Gravity Balancing Method for Limited-Power Flight Simulation M. Idan and A. Grunwald	77
AIAA-99-4030	Collaborative Decision Making Airspace Visualization Tool J. Krozel, D. Schleicher, and D. Dow	84
AIAA-99-4031	A Flight Simulator Study of Enhanced/Synthetic Vision for Helicopter Search and Rescue—Systems Development and En Route Objective Performance D. Sattler and L. Reid	99
AIAA-99-4032	A Design of Computational Environment for the HSE (Human-Sensibility Ergonomics) Simulator S. Yoon, C.-J. Kim, and S.-J. Park	122
Session 15-MST-3	**Simulation Testing, Verification, and Validation**	
AIAA-99-4033	Simulation for Acquisition, Validation and Electronic System Design G. Fitzhugh	132
AIAA-99-4034	A Methodology for Evaluating the Fidelity of Ground Based Simulators R. Hess and Y. Zeyada	NA
AIAA-99-4035	Image Dynamic Measurement System (IDMS-2) for Flight Simulation Fidelity Verification R. Lehmer and W. Chung	137

NA - Not Available

AIAA-99-4036 Determination of Navion Stability and Control Derivatives Using Frequency-Domain Techniques
W. Lewis and R. Catterall 144

AIAA-99-4037 Design and Flight Tests of a Ground Collision Avoidance System
G. Sachs, U. Sennes, and H.-L. Dudek 154

AIAA-99-4038 Use of the Cockpit-Transformed Simulator for Testing of Proposed Algorithms of Pilot's Control Actions During Aircraft Maneuvering Support System
O. Yakimenko and V. Demidov 160

Session 28-MST-4 Helicopter/Aircraft Modeling and Simulation

AIAA-99-4113 Spatial Frequency and Platform Motion Effects on Helicopter Altitude Control
J. Schroeder, W. Chung, and R. Hess 171

AIAA-99-4114 Dynamic Inflow Modeling for Simulation of a Helicopter Operating in Ground Effect
H. Xin, J. Prasad, and D. Peters 182

AIAA-99-4115 Simulation of Reconfigurable Heli-UAVs Using Main Rotor RPM Control in Failure Modes
I. Yavrucuk and J. Prasad . . . . . NA

AIAA-99-4187 Virtual Battle-Space Management System: A Real-Time Simulation Development and Management Tool
R. Subr 193

Session 29-MST-5 Network Technologies and Applications

AIAA-99-4116 DIS Real Time Exercise
R. Bertin . . . . . NA

AIAA-99-4117 ATN Application Development and Evaluation on ENRI ATN Simulation Testbed
K. Shiomi, K. Itano, and R. Otsuka 201

AIAA-99-4118 The Continued Evolution of a Real Time Air Combat Simulation Environment in a Secure Multi-Project Facility
T. Thomas . . . . . NA

Session 30-MST-6 Space Systems Simulation

AIAA-99-4119 High Fidelity GPS Satellite Simulator—A Training Tool
M. Baker 207

AIAA-99-4120 Dynamics and Control of a Novel Radio Telescope Antenna
M. Nahon 214

AIAA-99-4121 Flexible, Intuitive Interface in Computational Framework for Design of Complex Systems
N. Thompson and M. Tigges 223

AIAA-99-4345 Real-Time Simulation of a Free-Flying Robotic Vehicle
C. Carignan, J. Lane, and D. Akin 229

Session 43-MST-7 Low Cost Simulation, Device Development, and Environments

AIAA-99-4182 The "New" Air Combat Modeling, Simulation, and Analysis Federation at Wright-Patterson AFB, OH
S. Rodgers, G. Hellmann, T. Menke, D. Mittlesteadt, J. Zeh, and D. Richart 238

AIAA-99-4183 Low Cost Fast Trainer Prototyping via Distributed Simulation
L. Pollini, M. Innocenti, and E. Ronchieri 247

AIAA-99-4184 Hybrid Flight Simulator Based on Unmanned Aerial Vehicle and Autonomous Flight Situation Modeling
S. Sadovnychiy, I. Burdun, V. Krivtsov, and T. Ramires 257

AIAA-99-4185 Combat UAV Real Time SEAD Mission Simulation
P. Smith . . . . . NA

NA - Not Available

AIAA-99-4186 Experiments and Problem Issues in Simulating Real-Time Person-In-Loop Air-To-Ground Radar Imagery
K. Spuhl and D. Schmidt 268

AIAA-99-4188 An Overview of Atmospheric Modeling for Aeronautical and Aerospace Vehicles Simulation Applications
W. Vaughan, D. Johnson, and L. Ehernberger. 276

AIAA-99-4346 Low Cost, Real-Time Simulation of an Unmanned Aerial Vehicle
M. Drews and D. Forman 284

Session 44-MST-8 Aircraft Dynamics, Modeling, and Performance

AIAA-99-4189 Constrained Optimal Estimation
K. Choi and D. Mook . . . . . NA

AIAA-99-4190 Random Choice Model for Thrust Profile
W. Leite Filho and P. Pinto 293

AIAA-99-4191 AV-8BII Device S2F176 Flight Model Development Using a PC-Based Simulation Environment
D. Gingras . . . . . NA

AIAA-99-4192 Integrated Flight Mechanics and Aeroelastic Aircraft Modeling Using Object-Oriented Modeling Techniques
G. Looye 296

AIAA-99-4193 Development of a Free-Flight Simulation Infrastructure
E. Miles, P. Davis, and D. Wing 307

AIAA-99-4194 Aerodynamic Database Issues for Modeling Close Formation Flight
J. Myatt and W. Blake. 317

AIAA-99-4195 Simulation of Aircraft Emergencies by Means of Dynamically Similar Models
S. Sadovnychiy, A. Betin, and A. Ryshenko. 328

Session 45-MST-9 Simulator/Flight Training Devices and Program Overviews

AIAA-99-4196 Multi-Purpose Domed Simulator
W. Edmiston and J. VanHoogstrate Sr. 335

AIAA-99-4197 Automatic Correlated Terrain Database Generation and Management for Ground Vehicle Simulators
Y. Papelis, S. Allen, and B. Wehrle 343

AIAA-99-4198 The KARI Engineering Simulator: A Design Tool for Regional Transport Aircraft Development
K. Ro, S. Lee, I. Kim, V. Povstyanoy, and C.-H. Lim 351

AIAA-99-4199 Recent Improvements to the Large Amplitude Multi-Mode Aerospace Research Simulator (LAMARS)
B. Stadler. 362

AIAA-99-4200 Influence of Position Sensor Properties When Using a Perspective Flightpath Display
E. Theunissen and R. de Bruijn 372

AIAA-99-4201 Virtual Proving Ground Technical Development for Interactive and Collaborative Research and Engineering Design Studies
J. Turner, D. Solis, and Y. Papelis. . . . . NA

AIAA-99-4202 Language-Based Modeling and Simulation in Verilog-A: A Case Study of Gimbal Systems
R. Ahmadi, R. Smith, L. Rubin, and A. Zarkesh. 383

Session 69-MST-12 Vehicle Dynamic Modeling

AIAA-99-4320 Investigation of Dynamic Structural Models Suitable for the Simulation of Large Aircraft
W. Hashii, A. Munger, E. Elghandour, F. Kolkailah, and D. Biezad. 390

NA - Not Available

AIAA-99-4321 Flight Simulation and Control System Design Software for Prototype Development
K. Ro, M. Demyanov, J. Park, S. Go, and C.-H. Lim 400
AIAA-99-4322 Standard Airframe Model for Missile Autopilot Design and Evaluation
H. Eguchi, K. Obana, and M. Kiso 412
AIAA-99-4323 Hardware in the Loop Simulation of an Attitude Control System
W. Leite Filho, L. Mallaco, and D. Carrijo 423
AIAA-99-4324 Ground Effect Modeling Using a Hybrid Approach of Inverse Simulation and System Identification
D. Fischenberg 427
AIAA-99-4325 Nonlinear Dynamic Simulation of a Tethered Aerostat: A Fidelity Study
S. Jones and L. Schroeder 435
AIAA-99-4326 Hypervelocity Vehicle Modeling and Simulation Using GENESIS
P. McKeehen 442
AIAA-99-4327 Influence of Landing Gear Design on Helicopter Ground Resonance
I. Sharf and L. Monterrubio 452

Session 70-MST-13 Motion Systems, Human Factors, and Cueing

AIAA-99-4328 Developments in Human Centered Cueing Algorithms for Control of Flight Simulator Motion Systems
R. Telban, F. Cardullo, and J. Houck 463
AIAA-99-4329 Motion Fidelity Criteria for Roll-Lateral Translational Tasks
J. Mikula, W. Chung, and D. Tran 474
AIAA-99-4330 Motion Fidelity Criteria Based on Human Perception and Performance
A. White and V. Rodchenko 485
AIAA-99-4331 Evaluation of a Dynamic Seat and a Helmet Loader in a Motion-Based Flight Simulator
A. Rutten 494
AIAA-99-4332 A Description of the Cockpit Motion Facility and Research Flight Deck Simulator
R. Smith . . . . . NA
AIAA-99-4333 Preliminary Investigation of the Motion Fidelity Criterion for a Pitch-Longitudinal Translational Task
D. Tran, W. Chung, and J. Mikula 503
AIAA-99-4334 Acceleration Perception
L. Zaichik, V. Rodchenko, I. Rufov, Y. Yashin, and A. White 512
AIAA-99-4344 Advanced Numerical Techniques for Hi-Fidelity On- and Off-Road Vehicle Simulation
D. Solis, E. Haug, and S. Shome 521

Session 71-MST-14 Simulation Standards and Reuse

AIAA-99-4336 Use of the Mediator Design Pattern in the LaSRS^{++} Frameword
K. Cunningham 531
AIAA-99-4337 A Software Architecture to Facilitate Rapid Development of Real Time Simulation
E. Dillaber . . . . . NA
AIAA-99-4338 The Use of Multiple Threads in an Object-Oriented Real-Time Simulation
D. Geyer 538
AIAA-99-4339 A Generic Object-Oriented Implementation for Flight Control Systems
P. Glaab and M. Madden 547
AIAA-99-4340 Using Abstraction to Create a Portable Object-Oriented Simulation
P. Kenney and D. Geyer 554
AIAA-99-4341 EDCP—The Use of Object-Oriented Design in an Embedded Environment
R. Leslie . . . . . NA

NA - Not Available

AIAA-99-4342 An Object-Oriented Design for Rapid Modification of Filters
M. Madden 563
AIAA-99-4343 Challenges in Integrating Commercial Off-the Shelf (COTS) Software Into A Real-Time Simulation Environment
J. Nalepka, D. Caudill, L. Mutschler, C. Buell, L. Garcia, and D. O'Quinn . 572

NA - Not Available

Rapid Prototyping Tools Applicable to Avionics Requirements Development and Real Time Manned Simulation

C. Mike Bendall
The Boeing Company
St. Louis Missouri

William J. Bezdek1
The Boeing Company
St. Louis Missouri

Derek Green
The Boeing Company
St. Louis Missouri

Theresa Trapp
MDTSC
St. Louis Missouri

Abstract

Often times, the selected changes approved for aircraft development are turned into a list of requirements, the software designed, coded, tested, integrated with the hardware, ground tested, and finally flight tested. Different computer languages and hardware are often used during different phases of the development process. The objective of this paper is to show that by using several analytical development tools that all employ the same baseline software, the capability is provided to generate high quality code faster; provide software developers with a better understanding of the requirements; allow the customer / user / developer to gain a better insight into the avionics and displays being developed; and provide a significant impact to the development time and testing costs. The three tools are: an avionics test station, the Desktop Test Environment (DTE); a graphical display generation tool, VAPS from Virtual Prototypes; and a real time manned simulator, the Reconfigurable Manned Interactive Crew Station (RMICS). The tools were developed for separate purposes, but in combination, provide a synergy that can have a significant impact on the development cycle and support costs. This paper describes the tools, the interfaces, combining of the tools, and future work.

Introduction

The Desktop Test Environment (DTE) integrates the Operational Flight Program (OFP), the displays using VAPS from Virtual Prototypes, and models to be tested early in the design cycle. By adding a Reconfigurable Manned Interactive Crew Station (RMICS), personnel in the development and validation process can improve the overall capability by evaluating the design in a real time operational scenario.

---

1 Senior Member

Copyright © 1999 by The Boeing Company. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

The personnel, including the: operators / pilots, requirements engineers, software developers, display developers, and simulation engineers, can use rapid prototyping tools that are applicable to avionics requirements development, prototyping, and real time manned simulation.

Models needed to support avionics development for a fighter aircraft often consist of the:

- Airframe including the aerodynamics, flight controls, engine, weight and balance,
- Equations of motion,
- On-board navigation sensors, including the Inertial Navigation Set (INS), Radar Altimeter, weather or terrain following radar, etc.,
- Weapon system, including the weapons management computer, programmable armament control set, release racks, etc.,
- On-board weapon system sensors including radar, targeting forward looking infra-red(FLIR), radar warning receiver (RWR), etc.,
- Weapons,
- Air and ground targets,
- Other friendly assets, the communications and command and control structures connecting those assets to the system being developed,
- Environment including terrain, weather, and earth,
- Ownship displays, and the
- Central or Mission Avionics, often referred to as the Operational Flight Program (OFP).

All models need to communicate with each other using a virtual interface, similar to the real system, thereby providing the highest degree of certainty that the real system will perform as desired.

Weapon system models have evolved from specialized models designed and coded for each application to reusable models that can be utilized within multiple applications. Object-oriented design techniques and common interfaces that closely approximate "the real world" are often used. "Legacy" and /or government models are often "wrapped" to be integrated in a realistic manner.

The tools used for rapid prototyping of displays and avionics, the benefits gained, and the lessons learned will be discussed in the following paragraphs.

Previous Experience

A typical Operational Flight Program (OFP) development process has the requirements engineers, customer (pilots, operators, etc.), and simulation engineers develop several options for evaluation. These concepts were often coded in FORTRAN, the displays developed in OpenGL or a specialized graphics language, and the scenario presented with new or upgraded threats. The customer then evaluated and commented on the different options and a requirement list was made. These requirements become the basis for the software requirements that the OFP software development team uses while developing and testing the software. Not until the end of that process, which typically takes over a year, does the customer get to see the resultant code in the Manned Flight Hardware Simulator (MFHS). Avionics development begins with candidate concepts. The software concepts are designed, coded, and tested. If problems are found at that time, an extensive period is required to make changes and corrections. A typical process is shown in Figure 1.

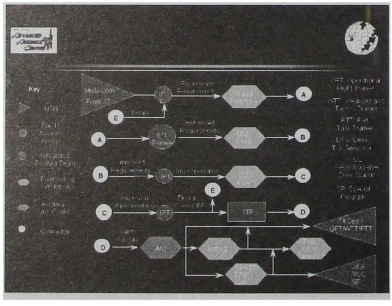


Figure 1 Typical OFP Development

Display Development

In the past, most display processors required dedicated code applicable only to that hardware set; therefore, the displays were hard coded and not reusable.

Recently, graphics standards like OpenGL, DirectX, etc., have appeared and are supported by many NT and UNIX workstation vendors. Unfortunately, onboard displays are still specialized for a particular aircraft and development environment. Today, several graphics boards support the use of higher level, graphical standards for the simulator, trainer, and aircraft.

Work on the Delta Clipper experimental rocket, shown in Figure 2, set the stage for automatic code generators for flight hardware computers. A commercial design tool, MatrixX, was used first to develop the flight control design. This success gave engineers confidence in machine generated application code.

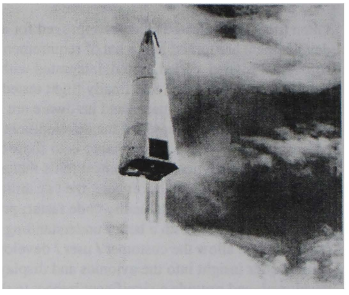


Figure 2 Delta Clipper

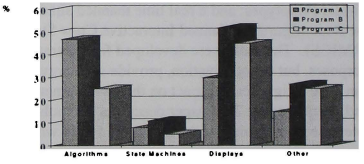


Figure 3 Typical breakdown for Embedded Software

As shown in Figure 3, comparative studies indicated that the display software area, which normally is similar in size to the underlying software design, would also greatly benefit from automatic code generation. For the algorithm software, the MatrixX experience was directly applicable.

Software Testing

Traditionally, testing avionics software begins in the laboratory. Expensive, near "flight quality" avionics

hardware and high-powered computers are brought together to provide the framework for the testing environment. When possible, real hardware components are used, but software models could easily replace them, thus reducing the development time and cost.

More recently, the trend has been toward increased use of software models of the hardware boxes. Software simulations of hardware have proven to be an effective solution to controlling testing costs.

The DTE has built on this concept and taken it one step further by integrating several existing pieces of software to form a virtual test system. The virtual test system concept allows the test system to be available on every developer's desktop allowing each to test the design frequently during development. By testing earlier in development, errors are detected and corrected earlier, making them less costly to fix overall.

Rapid Prototyping Tools

The key to rapid prototyping and development is to combine existing tools that developers and requirements personnel currently use. These include the DTE,VAPS, and RMICS. These assets are combined using application program interfaces (API)s to link them together and provide a rapid prototyping tool.

Desktop Test Environment – DTE

The purpose of the Desktop Test Environment (DTE) is to provide software developers the ability to test and debug the OFP at their desktop on a compatible PC running Windows NT. It allows them to test and debug their code rapidly at their desktop without using valuable laboratory time and hardware assets. It also facilitates execution of the OFP on the PC by creating a simulated environment using fighter aircraft models. (Note: it is intended for these models to be suitable for execution in both the PC and SGI environment). The DTE also simulates 1553 hardware drivers that allows data transfer between the OFP and aircraft models. Display simulation is accomplished by creating bezels and channel interface code in VAPS. DTE VAPS code is wrapped around VAPS display code created by the aircraft operator team. The DTE interface permits developers to control and manipulate aircraft models via a custom user interface. When operating within the DTE, Microsoft Studio Debugger can be used to perform OFP debugging for internal variables and algorithm analysis.

Requirements

Requirements for the DTE are determined by examining the use of the current Software Test Facilities (STF) and developer needs. Current PC limitations are also taken into account. The satisfied requirements include:

Available to engineers all of the time - The DTE is available on any PC with a minimum hardware requirement of a Pentium Pro 200, with 96 Megabytes of memory, and a sufficient hard drive to contain all executables. For full debug capability, the PC should have the Microsoft Studio Debugger installed. The DTE is capable of operation without the Studio Debugger, however the capabilities provided by the Studio Debugger are not available. Single or Dual processors are supported.

Allow execution of Studio Debugger - The DTE is capable of being used with Microsoft Studio Debugger. Studio Debugger provides the user the capability to set/clear breakpoint, single step the program execution, view class information, as well as many other capabilities explained in Studio Debugger documentation.

Ability to view 1553 Mux variables - The user is able to view external variables crossing the multiplex (Mux) bus using Common Test Language (CTL) and the Rapid Access Table (RAT). The display interfaces provide the user the capability to view and set model internal values as well as provide the ability to override model values. The user can readily save and load configuration files to assist in setting up the simulation environment.

Manipulate and view cockpit displays - The DTE has a VAPS wrapper that is compiled with the VAPS code. The wrapper provides display bezels and push buttons to the OFP VAPS code. Both fore and aft display sets are available as well as a simulated stick and throttle interface to control aircraft flight.

Manipulate aircraft models and control model inputs - Along with the capability to view external variables, the user can select aircraft configuration parameters and the suite of aircraft models that will run. In the initial DTE configuration, the user can only select a subset of the total models currently available on a full fighter configuration of the aircraft.

Allow free cycling of the OFP - The DTE runs the entire aircraft system as well as debuggers and other tools on a single CPU machine. This requirement is understood to mean that the DTE will appear to operate in real time to the OFP. The DTE is tied to the

infrastructure to allow controlled cycling of the OFP. By controlling the cycling of the OFP, the synchronization of all aircraft simulations can be accomplished.

Provide a common test environment - The DTE core and associated aircraft models function on both the desktop NT environment and in the STF-UNIX environment. They provide a common working environment for software engineers, therefore reducing the learning curve. Code is common for both environments, thus reducing the development and maintenance cost.

Components

The DTE is made up of a collection of components, many already in existence or modified to work together as shown in Figure 4.

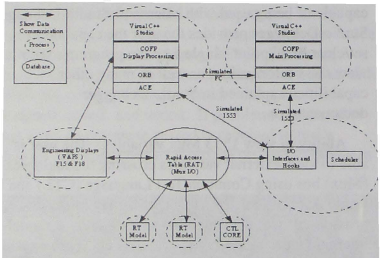


Figure 4 DTE Components

CTLS – Common Test Language System - A Boeing developed tool, has long been the standard tool for avionics software testing. Users are familiar with its capability and the tool is already cross platform capable. CTLS allows users to set and test external labels and MUX traffic.

RAT – Rapid Access Table - Another Boeing product, the RAT, acts as the shared memory pool for both the models and the OFP. It is also useful in testing the interface with the OFP. Along with the RAT, exists a library of template functions that allows easy use from within executables.

VAPS - Visual Application Builder Prototype System - VAPS allows real display code to be tested and run in conjunction with the DTE core. It is used to create the cockpit and the associated display formats. See the next section for more detail.

Scheduler / Time control - A Boeing product designed to control the cycle of the OFP and the associated simulated hardware. The scheduler forces processes to run in the correct order and scales time to prevent frame overruns.

Simulated 1553 – The simulated 1553 acts like the 1553 Mux Bus and transfers necessary data from the simulated models to the OFP via the RAT. The data is viewed using the RAT or the simulated MUX monitor.

All of the components are available on the monitor to aid the engineer / tester to operate the simulation, including the cockpit display, debugger, simulated controls, etc. as shown in Figure 5.

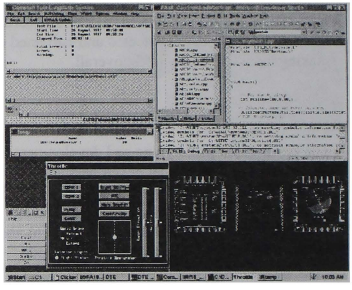


Figure 5 DTE Display Format

Visual Applications Builder -VAPS

VAPS is a tool for designing, testing, and implementing User Interfaces. By selecting basic or primitive geometric shaped building blocks, the desired graphical interface can be drawn. While these primitive shapes have some basic attributes like line style and color, they must be turned into VAPS pre-defined objects to permit them to assume the desired behavior. VAPS provides the ability to hierarchically combine objects to achieve more complex effects at run time.

VAPS allows for the drawing of the user interface with rapid access to animation and control of the items drawn. The movement of items on the screen may be controlled directly or through shared memory methods.

Full control of the user interface is based on the principles of the Finite State Machine model. Events can be detected, others may be generated, and symbols or whole groups of symbols can be controlled.

The steps towards providing an entire graphical, visual package are as follows:

- specify the look and basic behavior of objects,
- specify data flow between objects,
- define the application's behavior, and
- interact with the running application.

This separation of graphical representation from animation and control aspects allows for an easy graphical approach. A desired graphical change can be made using a point and click approach to change the object's behavior without having to consider each component in the object. The graphical object, often the more volatile portion of the interface, can be modified simply with no changes required to its animation.

The software generation feature of the VAPS tool provides a way, after design of the human machine interface, for the application to be rehosted to another platform. The open-ended feature of VAPS allows easy integration of programming for added complexity by providing access to the entire C programming language.

At Boeing, the project began with a proof of concept demonstration in a laboratory environment with an off-the-shelf graphic generator. After this, the generated code was then used on desktop PCs. Finally, this code was ported to the target processor. This allows incremental testing of actual display code that would eventually run on the target computer.

Traditionally, an independent laboratory group would have verified operation with display code generated for the laboratory test. Then another group would create a user interface to allow early testing on the desktop environment as shown in Figure 6, the Display Development Process. Finally, at the last step, the actual display code is verified on the target computer. However, one group as part of the development process can do all of these functions.

An additional benefit of this concept is the ability to provide early customer participation at a very low cost. Travel costs for the customer is zero. The executable was delivered to the customer to be run at their convenience. Quality improvements include more customers participating in the evaluation, a more thorough evaluation, and no confusion on the implementation. And again, this is the actual code, which is used and built to execute on the host platform.

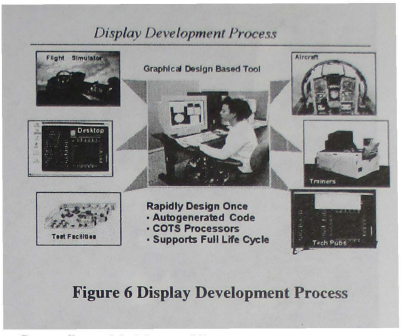


Figure 6 Display Development Process

Reconfigurable Manned Interactive Crew Station - RMICS

The Reconfigurable Manned Interactive Crew Station (RMICS) provides a more affordable simulator to prototype new concepts, do requirements trade-offs, develop/test prototype displays, checkout software modifications, and provide early real time testing. Several RMICS are currently in use at Boeing and have been delivered to China Lake and other locations where they are used for engineering development, prototype testing, and operational analysis. In addition, the RMICS is the type of simulator that makes an excellent platform for checkout of the Operational Flight Program (OFP) and associated controls / displays in realistic scenarios. Figure 7 shows the basic layout and features.

The features of the RMICS include the capability to use one or more graphics processors. It can be configured with the split screen display as shown in Figure 7. Other configurations include using a separate cockpit display with one 27 inch CRT for the cockpit and another mounted above for the HUD/Out-the-Window (OTW) channel. This configuration allows the cockpit to be used in operational scenarios by pilots, which represents the actual configuration, size, and layout long before any hardware is built. Other configurations include a low-end desktop environment with a single 19-inch monitor split screen display or a single cockpit CRT with multiple OTW projectors.

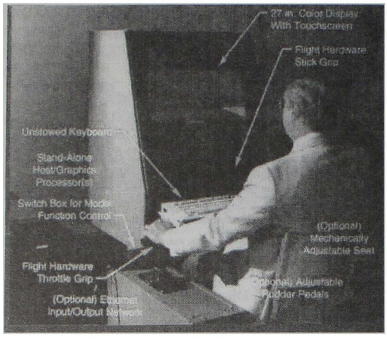


Figure 7 RMICS

The goals satisfied with the RMICS include displaying HUD, OTW, and cockpit on a single CRT, providing a station for early investigations of Pilot / Vehicle interfaces, provide pilot training, and providing networks of multiple aircraft. The RMICS is currently available with over a dozen cockpit displays and associated airframe / avionics configurations. This has allowed the RMICS to be used for generic, unclassified simulations as well as for specific programs by substituting the actual airframe, controls system, avionics, or displays.

Sub-systems comprising a generic RMICS are shown on the block diagram in Figure 8. It includes the basic enclosure and an image generator, like an SGI, which also serves as the host computer for the airframe and avionics models, and executive program.

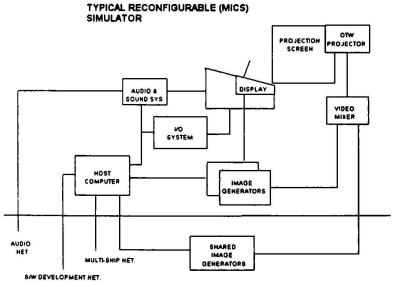


Figure 8 RMICS Block Diagram

The I/O System hardware often uses a PC with serial interface boards. It also houses the Ethernet interface and sound simulator. RMICS aircraft simulations can be run on a variety of Silicon Graphics computers, including 340 VGX, Crimson VGXT or newer Onyx.

The OFP, graphics, airframe, and environment are normally resident on the same host machine as the graphics, but can be distributed among several machines. Since the software and interfaces are common, higher fidelity displays, or other computers, may be added to the network.

The RMICS provides the capability to have dissimilar applications communicate over an interconnection backplane. It is possible to integrate time-based applications and event-based applications, to communicate seamlessly over the communications backplane. The configuration was first used for Industry / Interactive Training Systems and Education Conference (I/ITSEC) '93, where a RMICS interacted with a government developed air-to-air digital target model (Joint Modeling and Simulation Systems - JMASS[8]), as well as an off-board sensor system. All components in the simulation communicated over the Distributed Information System (DIS)[9] network with other DIS players. The RMICS software supports different interfaces to many "legacy" models, initially allowing them to use any format in the same simulation and converting from one to the other. For example, a message corresponding to the Boeing Flight Simulation Communications (COMM) array is used for the aircraft models, called Af110. During each update of Af110, the port is passed information about the state of the aircraft, including x, y, and z- geocentric coordinates, heading, and pitch, yaw, and roll angles. These separate items are packed into the message by the port and sent via the backplane to a separate task for conversion to DIS entity state packets.

Integration of Tools for Rapid Prototyping

The reconfigurable manned interactive crew station (RMICS) simulator used for this test is located in the Pilot Vehicle Interface (PVI) Center at Boeing Flight Simulation in St. Louis and runs on a Silicon Graphics platform. The Desktop Test Environment (DTE) executes on a 200 MHz Pentium Pro Windows NT based PC located on the same Ethernet network as the SGI Host program. A generic fighter cockpit running at a 60-hertz update rate was the crew station model selected. The first step in integrating the RMICS with the DTE was to select a message passing protocol. The RMICS host software uses the Unified Message

Protocol (UMP) developed by the Boeing Flight Simulation Technology group in St. Louis. This protocol provides inter-process and network communications via Berkeley Sockets. UMP libraries are available on a number of different platforms and operating systems, including SGI, Sun, Vax, and Windows NT. A unique port value was assigned for the message passing to be done between the host and the DTE. By specifying RECEIVER_CONVERT within the "receive" call and NO_CONVERT on the "send" side, the UMP software handles any data conversion that may be required when sending across different platforms. In addition, data types within the message allow different types of data to be sent within the same message (i.e., floats, integers, doubles, etc).

The initial message structure that was implemented sent the RMICS crew station data to the DTE. This included the stick longitudinal and lateral percent, rudder percent, and throttle percent values as well as various discrete switch positions (e.g., flaps, gear, speedbrake, etc.). Not all of this information was initially used by the DTE software, but the ability to send messages and have the DTE airframe code use these values was proven. Since the DTE was not executing at 60-hertz, the message passing code was set up to send the messages at a 10-hertz update rate so that the message queues would not overrun.

The next step involved creating an "external aero" class within the DTE that would accept the airframe values calculated by the RMICS host software. The software would utilize values received in a message that included the original crew station values along with the airframe variables (e.g., equations of motion, engine, and flight control variables) that the RMICS airframe code had calculated. These were stored in the DTE RAT, in the same locations and formats as though they had been computed on the DTE. The VAPS displays being drawn on the PC showed that the "external aero" values were indeed driving the OFP code residing within the DTE. The next step is to have the VAPS displays operating on the SGI, which will reduce the load on the DTE and show the display in the cockpit.

Rapid Prototyping with Combined Tools

A team of Boeing engineers representing Avionics Software, Avionics Integration, Avionics Analysis, the Avionics Integration Center (AIC), Flight Simulation, and Training Systems performed an investigation to determine the potential costs and cost savings associated with:

- Adoption of the common OFP as a baseline for project development, and
- Maximum use of the existing high level language OFP as a baseline for the next iteration.

There are several benefits to rapid prototyping with combined tools including:

1. Reuse of the application software in a workstation environment for simulators and trainers provides:
   - a baseline for prototyping the next version of the application software,
   - improved top level requirements,
   - reuse of the prototype application software to support detail design and testing,
   - quick transitions from the OFP development to a simulation environment for evaluations or demonstrations, and
   - reduced flight testing over time as this process shows that it can provide a high quality OFP with reduced testing.
2. Development and simulation of the mission computer OFP primarily on the desktop avoids problems resulting from limited laboratory resources for validating software.
3. Common models, scenarios, and test cases to support "real world" real time simulation prototypes, the DTE, STF, AIF, flight hardware simulators, and trainer environments, to provide higher quality, better tested, and more complete scenarios.
4. Reduced time to market by using the high level OFP as a baseline for prototyping new versions.
5. A competitive advantage for trainers reusing software and tools directly.

The resources required for an update has been shown to divide into three parts to support the:

- program office and requirements,
- government flight test center, and
- contractor for development, ground testing, and flight testing.

The primary savings the team found are distributed pretty evenly with reduced flight-test time as the most dominant. The potential saving over several updates on three different programs is shown in Table 1.

Table 1

Potential Cost Savings in %		
Phase	**Cost to Develop**	**Total Savings**
Requirements	0.2	4
Development / Soft. Test	0.3	2
Integration Testing + MFHS	0.4	2
Flight Test	0.0	8
Trainers	0.2	4
Total	1.0	**20**

The percent savings shown are for the contractor portion only, but, since the development time and the amount of testing will be reduced, there should be some level of savings on the government segments as well.

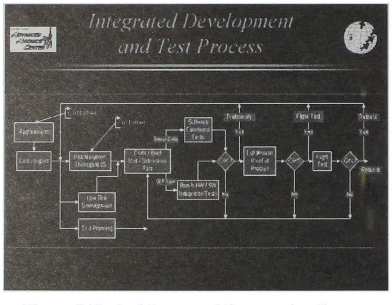


Figure 9 Typical Improved Prototyping Process

To achieve this level of savings, the development process should be updated to have an Integrated Product Team approach using common processes and tools as shown in Figure 9. This assumes the complete OFP has been coded in a high level language.

The requirement process can also be enhanced using the high level language OFP as part of the developments for the next iteration similar to what is shown in Figure 10.

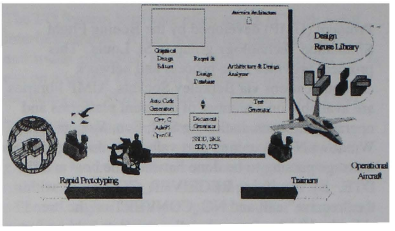


Figure 10 Requirements Prototype

This approach provides not only the benefits listed above, but allows the IPT and the customer / pilots / user / analyst to work together at both the beginning and during the development process. This allows changes, design decisions, and errors to be found using virtual operational mission scenarios during the requirements, development, and test process. By adopting some of the changes listed here , the resultant benefits should increase the savings to a higher percentage. The cost of implementing the change is relatively low, once the OFP is implemented in the high level object-oriented architecture.

Other Methodologies

There are other methodologies to provide a prototyping environment for avionics and display development. With the current trend in faster NT computers, it will be possible to provide a fairly robust prototyping tool with the combination of pilot controls, an improved graphics processor, a large monitor and a high-speed multi-processor NT computer configuration. There have been several initiatives to provide low cost flight hardware replica grips. Hardware supported OpenGL boards are available at low cost and 500 MHz Pentium III processors are commercially available.

In other cases, where additional out-the-window displays are needed, to develop a high off-boresight weapon, the existing development simulator can be updated to provide a prototyping capability using the actual OFP. As a matter of fact, the development simulator with the addition of the Operational Flight Program (OFP), like the one used in the DTE or STF can easily fulfill this role.

There are many tasks currently supported by development simulators, which are similar to those that

support flight hardware simulators, but the timing and purpose are often different. For instance, the customer / team integrates a new flight control concept into the development simulator with updated weapon launch algorithms and displays for evaluation. Once this new concept has been evaluated and becomes a requirement, actual changes are developed and tested as part of the updated software in the flight hardware simulator. New concepts are regularly integrated and tested in the simulators using project-generated floppy disks, which update/add new capabilities to a library, and flown in the simulators.

Development simulators are also being used to provide direct transfers from the flight control group to the simulator, where the control laws are designed in MATRIXx, auto code generated in Ada, electronically transferred to the simulator, and flown by project or customer pilots. A block diagram of the Engineering Simulator is shown in Figure 11.

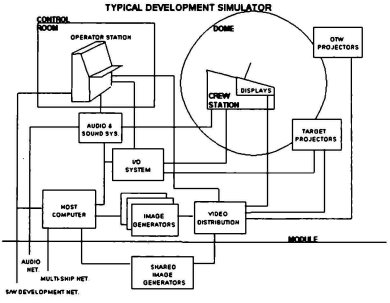


Figure 11 Development Simulator

The list of tasks performed in the Development Simulators include Flight Control Law Development, Training Of Operational Pilots, Training Of Engine Ramp Technicians, Support Of Foreign Military Sales and Training, Operational Assessments (Internal and External), Pilot Demonstrations, Flight Test Support, Pilot Proficiency Training, Classified Processing / Proprietary Programs, Development/Testing With New Threats Models, Cockpits For Other Programs, Marketing Support, and New Concept Development including Pilot / Vehicle Interface.

It is important to use a spiral design methodology, so the modeling and simulation components that support system development and testing activities can be shared by several development and testing environments, as shown in Figure 12.

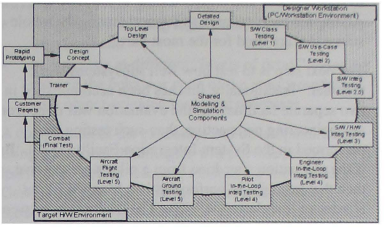


Figure 12 Test Cycle

Testing

Customer requirements need to be well understood and prototype design concepts developed and tested. Responsibility for developing the design concepts should be assigned to an Integrated Product Development (IPD) team, which is composed of representatives from several development and testing disciplines. The IPD team should proceed from prototyping design concepts by combining the DTE, RMICS simulator, and the customer to look at different options and refine the requirements. Design and initial testing of embedded software will take place on the DTE in a PC environment.

Detailed design / coding continues in the PC environment. Initial software testing (class testing) also is accomplished in the Software Development Facility (SDF) by augmenting the tools necessary to facilitate software design and construction with tools that facilitate object-oriented class testing. Intermediate software testing (i.e., component testing and software integration testing) is accomplished in the Desktop Test Environment (DTE), which also leverages the standard PC/Workstation configuration with additional tools tailored to support such testing activities and can be quickly moved to the simulator for additional pilot/customer evaluation.

At this point, a transition is made to test on the target hardware. Traditionally, software / hardware integration testing of the Mission Computer subsystem is accomplished in a Software Test Facility (STF). To verify that functionality has not been lost in the

transition from host to target hardware, some of the testing accomplished on the PC/Workstation configuration can be repeated using the same tools used to support class and cluster testing in the DTE. Additional tools tailored to support testing the whole subsystem are used for the most part.

The next step is to test how well subsystems work together. Traditionally, this has been accomplished in two separate facilities. If the evaluation was done from an engineering perspective, then such testing was performed in the System Integration Facility (SIF). But if the evaluation was done from a pilot's perspective, then that testing was performed in a Manned Flight Hardware Simulator (MFHS). The distinction between these two testing environments is now blurring. In the future, it will be quite possible to run some pilot evaluations in the SIF, which will have some Out-the-Window (OTW) display capability, which is necessary to support that testing. Only when the pilot wishes to evaluate how the avionics system provides feedback on aircraft flying qualities is it essential to perform tests in the MFHS. Tools to support testing in these environments are similar to those used to collect flight test data.

On-aircraft ground testing and flight testing is conducted to ensure that customer requirements have been satisfied in the production aircraft / weapon system and that the aircraft is capable of performing all assigned missions when flown by operational test pilots. Flight instrumentation systems monitor the interaction between subsystems as the pilot performs tests that are representative of various combat phases, and other airborne and ground-based monitoring systems establish actual performance of the aircraft / weapon system under test.

Trainers are also designed to closely replicate how the aircraft / weapon system will perform assigned missions, however testing tools are now designed to measure the performance of the trainee rather than the aircraft / weapon system. The trainer itself very much resembles a high fidelity Development or Manned Flight Hardware Simulator and shares many of the same modeling and simulation components used in that environment.

Conclusions

This paper described tools that can be used during various development and testing phases in the aircraft design and development process, interfaces that must be developed, and a spiral design approach that allows the same equipment and software to be used for requirements prototyping, detailed design, integration, and testing. The tools can be utilized to provide a requirements prototyping environment that:

- Improves the quality of the requirements by having the customer more closely involved,
- Improves the quality and reduces the development cycle by having the intermediate version of the software flown and evaluated by the customer / pilots, and
- Reduces the testing requirements in the STF and/or flight test by allowing the customer to evaluate the complete weapon system in realistic scenarios during development.

The Desktop Test Environment (DTE) tool was developed to support the testing of avionics software and displays at the engineer's desk. It runs at update rates to allow the developer to view the results and provides an increased number of low cost stations to test and integrate the Operational Flight Program (OFP) code and displays. The DTE environment is PC based and uses high level languages and a Windows NT operating system.

By providing developers with the ability to test their code during the entire development process, software reliability goes up and development costs go down.

The DTE tool has been used with great success on current programs and will continue to play an important role in future programs. Integration with the RMICS facility is just another step in the ability to use common avionics software / hardware throughout the development cycle.

VAPS allows early development of the displays using a graphical design tool. These displays can be quickly integrated into the DTE or the RMICS.

The Reconfigurable Manned Interactive Crew Station (RMICS) simulator consists of flight hardware grips and a single display surface showing the virtual cockpit layout with embedded avionics displays. Operational pilots have flown the simulator in real time during the requirements development phase using the RMICS and/or other types of simulators. The prototype avionics and weapon system changes can be rapidly tried on a version of, or an extension to, the avionics baseline.

By combining these tools and using common interfaces, the:

- avionics software developers,
- display developers,
- simulations engineers,

- systems engineers, and
- requirements engineers,

can work directly with pilots and can cost effectively develop prototype design concepts that serve as a starting point for actual avionics design changes. These tools provide the capability to generate higher quality code faster, provide software developers with a better understanding of the requirements, allow the customer / user / developer to gain a better insight into the avionics being developed, and provide the potential to significantly impact development and testing costs.

Future Applications

There are several next steps needed. VAPS and Boeing are working on a new version of VAPS which should be more applicable to flight hardware and the real time simulator. There are also better graphics cards that support OpenGL in hardware. That should also increase the speed of the graphics, which take up approximately a third of the CPU time. There may also be advantages to having the OFP supported on other processors, like the DEC Alpha or SGI. In some cases these processors already have software models which run faster than the NT computers or are desired for a trainer application. There are also additional topics desired for the avionics design and testing, updated interfaces, and the use of "legacy" models.

Avionics Design and Testing

The architecture used for object-oriented models can also be applied to flight hardware in building embedded software for flight. Many of the same design principles apply as well as the desire to be able to develop on one platform and then move the code to different target hardware.

In addition, it would be useful for the software developed for a simulation to be the same as that used in flight. There are several additional features that are needed for simulation software, like initialization in flight, slewing to different areas of the world, stopping, and starting the simulation, that is not normally included in flight software. With the object-oriented principles and the use of polymorphism, a flight hardware load could be very similar for the development, simulation, trainer, and flight hardware version. If successful, it would have a significant impact on overall program affordability.

Legacy Models

The models for the simulation can be converted in two ways using:

- wrappers on existing legacy models with common interfaces and
- new models at different "levels of detail" based on reusing the behavior code of the legacy models with object-oriented tools and processes.

A comparison of the execution and development time using the two methodologies will provide lessons learned, as well as working models, for the simulation. Generating interfaces for different "level of detail" models will provide a baseline of how to change from one model detail to another during execution.

Interfaces

The current simulation used Unified Message Passing (UMP) for communications. There are extensions to UMP, like Object UMP being used on other simulations that are applicable. In addition, there are government standards like the High Level Architecture (HLA)[χ] and commercial standards, like Common Object Request Broker (CORBA) that are being used and may be applicable.

[α] Bezdek, W.J & Herr, C., "Use of JMASS as a Many-on-Many Engagement Model", AIAA 96-349 July 1996.

[β] Beavin, W.C, "Simulation Meets Information: Bringing Aircraft Modeling and Simulation into the Information Age", AIAA-97-3591, August 1997.

[χ] HLA Briefing, Dr. Judith Dahlman, December 1994.

AN AVIONIC SYSTEM MODELING FRAMEWORK

Roger J. Bluff
Sensor & Avionic Systems Dept,
DERA
Farnborough, Hampshire, UK.
email: rogerbluff@acm.org

Abstract

The complexity of evolving avionic systems and standards demands the need for cost effective risk reduction to facilitate the smooth transition into new technologies. Modeling and simulation techniques are a flexible and cost effective means of achieving this aim. This paper defines an avionic modeling framework for the construction of an avionic system model that can analyse the hardware and software comonents and their interaction in a system. Three modeling domains have been defined: behavioural, performance and visualisation. Behavioural models primarily constitute the software components of a system, the performance models analyse the hardware concepts, and visualisation models enhance the understanding of a proposed system's operation. The Shlaer-Mellor object oriented analysis (OOA) method has been used to develop a specification for the avionic modeling framework. The capabilities of the modeling suite are described including the models of the architecture building blocks of the framework. These include data communication networks, processing modules, memory nad input/output devices. It is discussed how architecture performance bottlenecks can be identified nad how the capacity of the system to support avionic functionality can be derived.

Introduction

In order to reduce the cost of future avionic systems developments avionic architectures and standards have been developed to encourage system component reuse and portabiliity. In the civil avionic community this has led to the development of the Aeronautical Radio, INC (ARINC) 650 series of avionic architectures and standards[1]. Their purpose being to define a compatible framework that airframe manufacturers and avionic suppliers can make reference to. A competitive source of supply is encouraged to ensure that the most cost effective solutions are adopted.

An important architecture concept that forms an integral part of the evolving avionic standards is the notion of an Integrated Modular Avionic (IMA) architecture. Individual architecture building blocks are defined that can be reused to form individual architecture instances and system implementations. These building blocks can include hardware components (processing modules, data communication networks) as well as components of the software architecture. ARINC 651 defines a set of rules and guidelines for the realisation of a civil IMA architecture[2]. A range of architecture options are described to support a potential system development.

In support of portability the 650 series of standards has defined an APplication/EXecutive (APEX) ARINC 653 that describes an interface between an avionic application and an underlying avionic operating system[3].

The instantiation of an IMA system also encourages the enhancement of fault tolerance and the improvement in the availability of avionic systems. The guarantee of a maintenance free operating period is an important objective for the airlines[4].

The development of these standards and their potential realisation in an avionic system development needs to be assessed in order to ensure that current and future operating requirements can be met. Simulation modelling and analysis is a flexible means of contributing to this evaluation since a range of application loads can be investigated through modelling.

This paper is concerned with the development of a modelling framework that can support the analysis of the ARINC 650 series standards to reduce the risk in their adoption in future avionic systems.

An important aim of modelling is to gain an understanding of a potential system's operation early in the design cycle, when changes can be made more flexibly and with less cost. Significant design changes later in a design cycle can become prohibitively expensive.

This paper is organised as follows. An Avionic System model framework is defined. Component models of this framework is then described including an ARINC 651 (IMA) performance model and an ARINC 653 software behavioural model. Visualisation techniques in support of the performance model is then briefly

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

described. A simple analytical model is then discussed to assist in the validation of the modelling framework. The construction of a confidence interval is discussed to enable a demonstrable avionic concept to be validated by the modeling suite. How a system model can be realised from the various modelling components is then discussed. Options for future work are then described. A mechanism for fault tolerance is then analysed to illustrate an application of the modeling suite.

Avionic System Model Framework

An avionic system model framework has been defined to formalise the relationships between different modelling components that a subset of which would be required to analyse a proposed avionic system. The Shlaer-Mellor Object-Oriented Analysis (OOA) method[5], has been applied to develop this definition. The SES/objectbench tool supports Shlaer-Mellor[6], and was used to capture the analysis information.

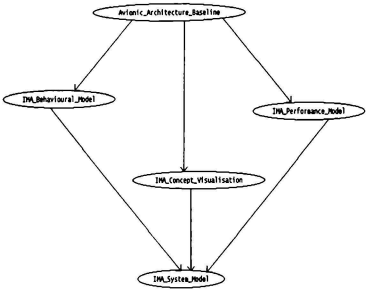


Figure 1: Avionic System Model Framework Domains

As shown in figure 1 above, the first stage in a Shlaer-Mellor analysis is to define the independent domains: each domain has it's own rules and policies that are common to the domain's objects. The behavioural, performance, and concept visualisation domains support the need to assess an avionic architecture. These domains collectively support the realisation of an IMA System Model.

The Avionic Architecture Baseline provides the modelling component reference for the individual modelling domains. Figure 2 below shows the individual subsystems defined within this domain. Each subsystem defines objects for the software architecture, the hardware components and facilities for the provision of fault tolerance.
Collectively these form the baseline reference.

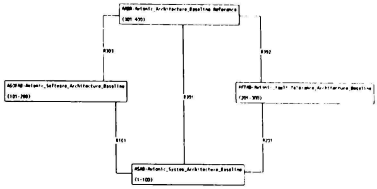


Figure 2: Avionic Architecture Baseline Subsystems

Within the Software architecture subsystem objects have been defined that formalises the relationships between the various elements of the system software. Figure 3 below shows a selection of these objects: instances of software interfaces are shown between avionic applications and instances of supporting hardware.

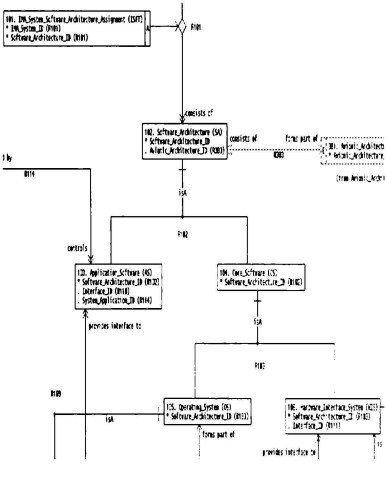


Figure 3: Software Architecture Objects

In a similar way the hardware components that instances of the software architecture will reside can be defined as objects as shown in figure 4 below. The partial range of Line Replaceable Modules is shown, a subset of which would reside in instances of IMA cabinets as defined in reference 2.

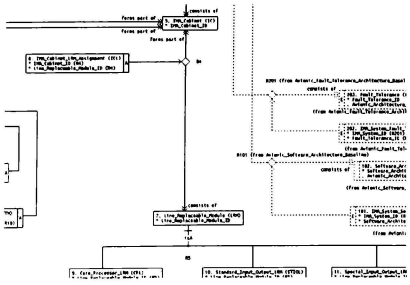


Figure 4: System Hardware Objects

Future avionic systems will need to progressively support more extensive facilities for fault tolerance. Figure 5 below specifies the range of techniques that will need to be supported.

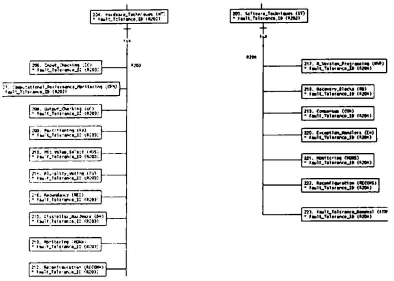


Figure 5: System Fault Tolerance Objects

It has been found a Shlaer-Mellor analysis provides a succinct means of specifying systems and their various options and instances. Consequently this method was found to be very suitable for describing a modelling framework for an Avionic System Model.

From the defined avionic architecture baseline modelling components can be identified for software and hardware elements of the system.

ARINC 651 (IMA) Performance Model

The System Hardware objects have been modelled as a system architecture hardware performance analysis model based on ARINC 651[2]. The SES/workbench tool[7], has been utilised to develop this model. This tool provides the capability to develop multi-purpose discrete event simulations. Performance statistics can be gathered reporting: means, maximum and minimum values and their standard deviation.

The main modules for the ARINC 651 performance model are shown in figure 6 below. The IMA_Architecture defines the avionic architecture to be modelled. The model can be configured to represent different architecture instance options, ARINC(2). Computing resources exits in the IMA_Cabinets; data can be transferred via the Bus; various aircraft sensors, actuators and interfaces have also been modelled.

A typical IMA_Cabinet is shown below in figure 7 below: three processors with access to a main memory module via a backplane bus have been modelled. The backplane also has access to the main data bus via a gateway module.

The performance model of a processor server is shown below in figure 8. Results from the processor can be sent to an output or written to memory. The "Write_To_RAM" modelling component allocates memory space from the "Local_RAM" resource pool (local processor memory). Alternatively the processor may need access to data from its memory or may need to delete unwanted data. These alternatives are shown

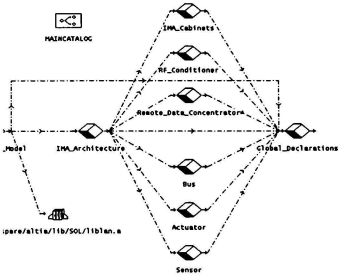


Figure 6: ARINC 651 main modules

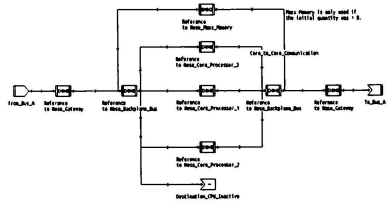


Figure 7: An Example IMA Cabinet

by the transaction arcs "Read_From_RAM" and "Release_Memory" respectively. Any access required to ROM is shown by the "ROM_Access" server associated transaction arc. Access to RAM is also modelled by a server ("RAM_Access") since memory is a finite resource that takes time to access.

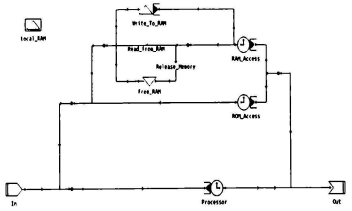


Figure 8: Processing Resource Performance Model

The processor server icon shown in figure 8, for example, models a queue and a delay. Different queuing disciplines can modelled; first come first served is commonly used. A range of probability distributions can be entered for the service delay, a negative exponential is often used for queuing systems. Processor response time, and queue length can be monitored. Process utilisation can be neasured and is defined as the proportion of time the server is busy and can be expressed as a percentage. Time to access memory can also be monitored. From these basic performance parameters, architecture performance bottlenecks can be identified and the system's capacity to support avionic applications can be derived. Continuously growing queue lengths found for a server, during a simulation, could indicate a bottleneck has been identified and potentially the capacity of the system will be exceeded to perform tasks. In this situation either the arrival rate of jobs needs to be reduced or the power of the server needs to be increased.

ARINC 653 Software Behavioural Model

A software behavioural of ARINC 653[3] has been completed based utilising the Shlaer-Mellor method. The first stage is to define domain chart as illustrated in figure 9 below.

By convention the top of the domain chart is the application, the middle the service and the bottom is the implementation or architecture domain. The interface between domains is an important area of development in the Shlaer-Mellor method.

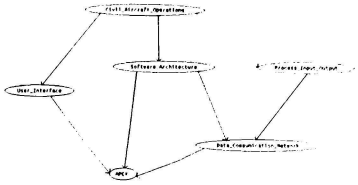


Figure 9: Software Behavioural Model Domain Chart

The operation of civil aircraft requires the support of an avionic software architecture, and in turn APEX the application domain, is an important component of the architecture. The individual subsystems within the APEX domain is shown in figure 10 below.

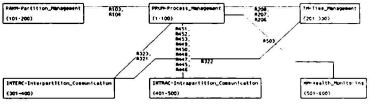


Figure 10: APEX Subsystems

Each subsystem contains a set of objects relating to service calls required by the functional groupings. As an example figure 11 below illustrates a selection of objects defined for process management. The process object is created, suspended, stopped for example by the surrounding service call objects. In the Shlaer-Mellor method roles as well as entities can be defined as objects[5].

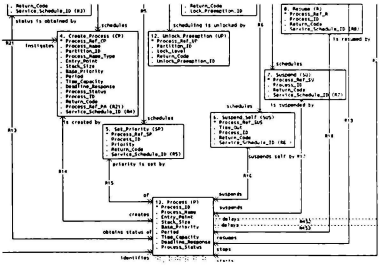


Figure 11: Process Management Objects

Each object has a lifecycle, figure 12 below shows the lifecycle for the Process object. The state transitions

can be modelled with the support of a discrete event simulator. The APEX model has been implemented using the SES/objectbench tool[6], simulation of the object lifecycles is supported. Delays for event transitions can be defined in a similar manner to the service delays that can be defined in SES/workbench[7], including probability distributions for example.

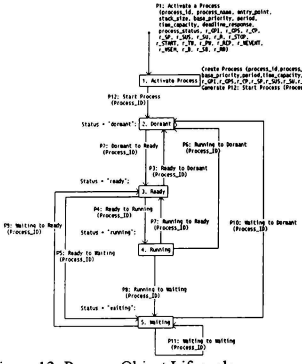


Figure 12: Process Object Lifecycle

An important aim of behavioural modelling is to understand the operation of a system and to ensure that the sequence of events occur as expected. Through simulation race conditions can be investigated. Figure 13 below defines the lifecycle of the Suspend-Self object. Shlaer- Mellor supports a timer mechanisms whereby events can be generated after a pre-determined delay. In the Suspend-Self lifecycle if the process can be suspended after a time out delay the process is resumed. The time out mechanism ensures that a process doesn't lock up. Any form of failure is reported in the return code.

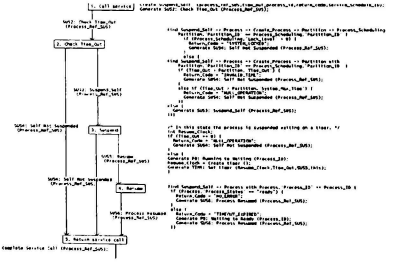


Figure 13: Suspend-Self Object Lifecycle

The investigation of the determinism of the behaviour of the object lifecycles is well suited to this form of simulation modelling. This form of analysis is particularly important for the development of avionic systems.

Visualisation

Visualisation displays have been developed to support the avionic modelling framework. Figure 14 below presents a graphic visualisation of a five cabinet avionic system ACR(Avionics Computing Resource)s including the processing resources, backplane buses and the main global avionic data bus. In this case the utilisation of the resources is being monitored during a simulation. Highly utilised resources are shown in a darker colour. Potential system bottlenecks could be readily appreciated with this form of visualisation.

The Altia tool has been utilised for the development of the visualisation displays[8]. The graphical display can be developed to react to an event generated by SES/workbench or SES/objectbench; data monitored by these tools can also be passed to a graphical display including on line performance statistics as illustrated in figure 14.

Visualisation displays are an important way of conveying modelling results to non-experts and can be very useful for generating customer oriented displays. In terms of an avionic system this could include an example cockpit display.

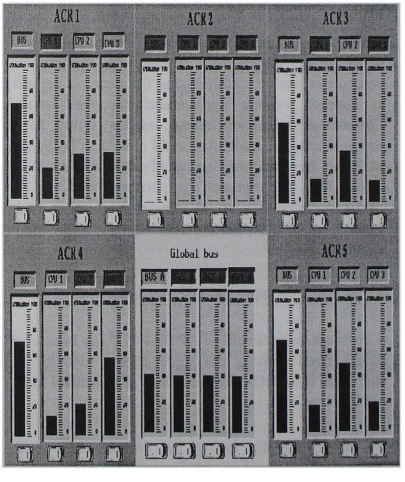


Figure 14: Processor Utilisation

Analytical Model

In order to assist in the initial validation of the modelling framework an analytical model has been developed. Service centres have been defined for the main components of the architecture components including: processors, backplane buses and main avionic data buses. The following inputs and outputs to the analytical model have been defined as follows:

Inputs:

K = number of service centres
k = 1,...,K
λ = arrival rate to system (=V/T)
v_k = number of visits to service centre
n = number of transactions in system
$V_k = v_k/n$
S_k = mean service time at service centre
T = model simulation time (in simulation units)

Outputs:

λ_{max} = maximum capacity of system
R_k = mean response time
X_k = throughput (or completion rate)
U_k = server utilisation
Q_k = mean queue length
N = mean number of transactions in system
R = mean system response time

From the above definitions the following equation can be derived from Little's Law[9]:

$$\lambda_{max} = 1/(\max_k\{V_kS_k\}) \quad (1)$$

if $\lambda \geq \lambda_{max}$ then insufficient capacity

$$X_k = \lambda \quad (2)$$

$$U_k = X_kS_k \quad (3)$$

$$V_bS_b = \max_k\{V_kS_k\} \quad (4)$$

Equation (1) derives the maximum capacity of a system to absorb tasks; equation (2) calculates the system throughput; equation (3) calculates the utilisation of a model component; and finally equation (4) derives the system's bottleneck.

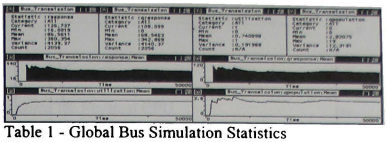


Table 1 - Global Bus Simulation Statistics

The above equations are valid provided the number of tasks entering the system equal those completing service. The avionic system model simulation was run until flow balance was achieved. Table 1 above lists the simulation statistics for the main avionic bus under flow balance conditions during an example analysis. Table 2 below uses these statistics to solve equations (1) - (4).

Model Component	Avionic Bus
Measured Values	
Number of visits (V)	2056
Mean response Time (rt)	86.5611 s
Mean queue response time (qt)	68.5463 s
Calculated Values	
Mean service time (S = rt - qt)	18.0148 s
Mean arrival rate (= V/T)	0.04112 transactions/ s
Throughput (X = λ)	0.04112 transactions/ s
Utilisation (U = XS)	0.7408

Table 2 - Global Bus Statistics

As can be seen the calculated Utilisation is in agreement with the simulation results. Since flow balance can be assumed the system bottleneck can be identified (equation 4) and the maximum capacity of the system can be derived (equation 1).

As the arrival rate at the system bottleneck increases (note: there can be more than one bottleneck simultaneously) the response time is bounded asymtotically as shown in figure 15 below. The points plotted were obtained from the above example simulation analysis.

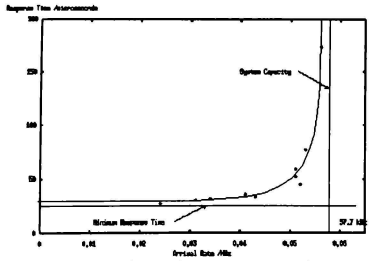


Figure 15: Effect of arrival rate on response time

Through the above analysis and the use of equation 1 the maximum sustainable arrival rate for the system can be derived.

Modeling Framework Validation

System performance measures from an avionic demonstration rig can be compared against corresponding simulation results from the models. The aim would be to determine whether the avionic modelling suite can be formulated to be a reasonably accurate representation of a demonstrable avionic concept.

One possible approach is to apply a statistical test to determine whether the underlying distributions of the data sets can be safely regarded as being the same. These tests assume the data is independent, identically distributed (IID) and for most real-world systems and simulations this assumption is not valid.

An alternative approach is to use a confidence interval. A hypothesis test will either be rejected or accepted. A confidence interval will also provide this information but will also provide the possible range of values for the parameter.

To determine the range in a measured parameter, probabilistic bounds can be derived. For a given parameter there is a probability of $1-\alpha$ that the parameter is in the interval (c_1, c_2):

Probability $\{c_1 \le \mu \le c_2\} = 1-\alpha$

The interval (c_1, c_2) is the confidence interval, α is the significance level, $100(1-\alpha)$ is the confidence level, and $1-\alpha$ is the confidence coefficient. Confidence levels are normally near 100%; while significance levels are typically near zero.

A validation exercise would ideally conduct n experiments on an avionic demonstrator and the simulation model. There would then be a one-to-one correspondence between the i th test on the demonstrator and the i th test on the simulation so that the observations will be paired.

The two data sets would then be treated as one sample of n pairs. For each pair, the difference in the performance measure would then be derived. A confidence interval would then be constructed for the difference. If the confidence interval includes zero, the avionic demonstrator and the simulation model would not then be significantly different. Hence, the model would have been validated. If there is a difference then the data would need to be examined and the model refined as necessary. The confidence interval test for the data set would then need to be repeated.

From the central limit theorem a $100(1-\alpha)\%$ confidence interval for a population mean is given by:

$$\left(\bar{x} - Z_{1-\alpha/2} \, s/\sqrt{n}, \; \bar{x} + Z_{1-\alpha/2} \, s/\sqrt{n} \right)$$

$\bar{x}$ = sample mean
s = sample standard derivation
n = sample size

$Z_{1-\alpha/2}$ is the $(1-\alpha/2)$ quantile of a unit normal variate. These values can be looked up from a standard table.

The above confidence interval can only be applied to data sets with large sample sizes, n>30. For smaller sample sizes the confidence interval is given by:

$$\left(\bar{x} - t_{[1-\alpha/2;n-1]} \, s/\sqrt{n}, \; \bar{x} + t_{[1-\alpha/2;n-1]} \, s/\sqrt{n} \right)$$

$t_{[1-\alpha/2;n-1]}$ is the $(1-\alpha/2)$ quantile of a t-variate with $n-1$ degrees of freedom. These values can also be looked up in table form. In a simpler form:

Confidence Interval for mean = $\bar{x} \pm t\sqrt{s^2/n}$

Where s^2 = sample variance.

The sample means can be derived from the full range of performance data measured from the demonstrator and the avionics modelling suite: response times for equivalent workloads for example.

System Model

A system model would aim to provide an analysis of the mapping of avionic applications and supporting system software across available hardware resources. In the case of the avionic system model framework, this would provide a link between software behavioural models and their associated hardware performance models.

Figure 16 below illustrates a simple object lifecycle that could be modelling a software thread. In State_2 an event is passed from SES/objectbench to an SES/workbench performance model along with two data items. In State_3 an event is returned from SES/workbench. In this case a parameter is passed between the models during the event transition and

defines the time delay for the state transition. A link therefore can be made between the behaviour of software and the associated performance implications. Instances of software objects can be mapped across supporting hardware for analysis purposes to form a system model.

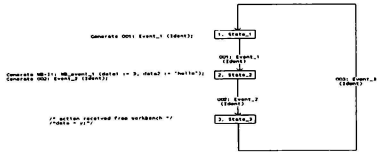


Figure 16: Software-Hardware Simulation Link

The performance analysis domain is best suited to modelling contention for resources. Figure 17 below shows the companion SES/workbench model that provides an event to the software model defined in figure 16. The "Write_To_Memory_Event" transfers resources from the Memory resource pool. The next icon "Event_Release" transfers resources back to the Memory resource pool and transfers a event back to the software model.

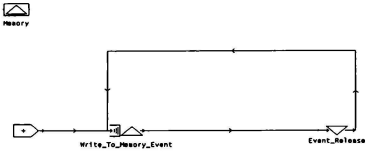


Figure 17: Hardware-Software Simulation Link

This combined model could represent software behavioural model accessing physical memory in the hardware domain. Figure 18 below represents a physical processor; an established link between the software and hardware modelling domains. An understanding can then be developed of the interaction between the hardware and software components of a system. How software is mapped across an architecture could be assessed in terms of behaviour and the performance implications. In general a system model should aim to provide an optimisation of the architecture concepts under assessment.

The developed modelling suite (ARINC 651 and 653) can be used as individual models or linked together as discussed to form a system model.

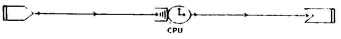


Figure 18: Physical CPU

Future work

Future modelling activities planned include the extension of the software architecture model to include the hardware support layer that exists beneath APEX and to develop a distributed executive above the application interface layer. A distributed executive is required to control the distribution of applications across the hardware resources of the entire avionic architecture. Figure 19 below illustrates the proposed extension to the existing model.

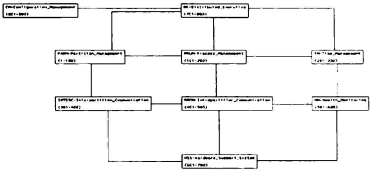


Figure 19: Software Architecture Model

Also shown in figure 19 is a configuration management subsystem that would assist in the fault management of a system. Figure 20 below illustrates this same subsystem modelled as part of the performance modelling domain.

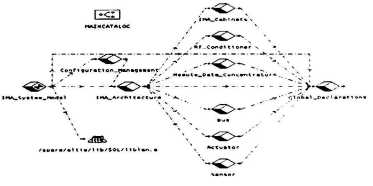


Figure 20: Avionic Architecture Performance Model

In addition to the further development of avionic modelling components it is planned to validate the modelling suite against an avionic systems demonstrator developed under the Control Technology

Programme[10]. A study of the ARINC standards was included in this project. The Avionic System Model will be formulated to represent the demonstrator. Similar experiments will then be conducted on the avionic rig and the model. Behaviour and performance results will be gathered and their data sets statistically compared. If they are found to be statistically equal then the modelling suite will validated against the rig; alternatively, model refinements will be made as necessary. The framework can equally be applied to military standards and architectures.

Fault tolerance

In figure 21 below a set of subsystems are defined for an IMA software architecture. Each subsystem contains objects pertinent to their respective subsystem. Each subsystem represents an important layer in an IMA system that would need to be analysed by the modeling suite.

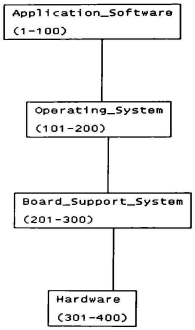


Figure 21: Software Architecture - Subsystems.

As an example of the types of objects that could be found in a subsystem, figure 22 below illustrates a NMR voting system. A software voting system would reside in the Application_Software subsystem and a hardware voting system would form part of the Board_Support_System subsystem. Instances of the process object could be associated with instances of the voter object. These simple objects could be used to model various voting architectures, an example voting structure is described in reference 11.

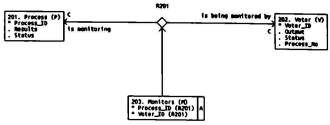


Figure 22: NMR Voter - Objects.

Figure 23 shown below defines the lifecycle of the process object. An instance of process is created and then waits for the availability of an instance of the voter object. Once a voter object has been assigned the result from the process is made available to the voter and once served the process instance is deleted. Each state transition shown can have a simulation time delay defined.

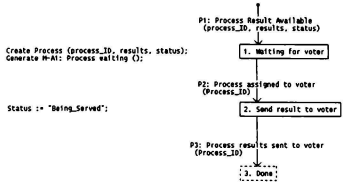


Figure 23: Process Lifecycle.

The voter object lifecycle shown in figure 24 below waits in an idle state until it is assigned to a process object. Once assigned the output from the process is read and the voter returns to an idle state, when it then becomes available for another process output.

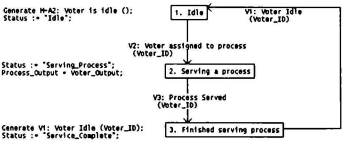


Figure 24: Voter Lifecycle.

The monitor object has two lifecycles: the first (assigner) associates an instance of a process object with a voter object and the second compares results from previous processes for the purpose of masking a process. A masked process is considered to be in error[3]. Figure 25 below defines the assigner lifecycle: an

instance of a process object that is idle and not currently being serviced is associated with a voter object that is also available.

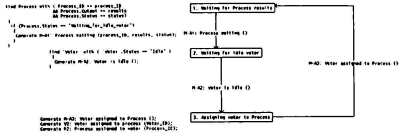


Figure 25: Monitor - Assigner Lifecycle.

The monitor lifecycle (figure 26) saves the output from a process and determines if it is consistent with previous process results. An attribute of the voter object determines the number of processes that the voter will monitor. At least two results need to be consistent before a process in error can be detected. Once a process is found to be in error this process is masked as shown in figure 26 below.

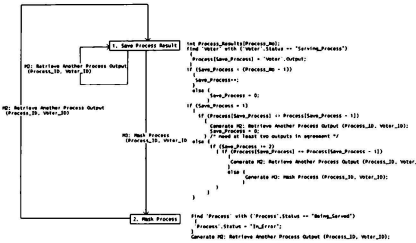


Figure 26: Monitor Lifecycle.

The above behavioural model could be used to analyse a proposed NMR architecture in support of a proposed IMA system. Since the model would support simulation, timing problems including race conditions could also be assessed.

CONCLUSIONS

A framework for the an Avionic System Model has been defined. ARINC 653 and 651 software behavioural and performance models respectively have been described. It has been shown how architecture bottlenecks can be identified and the system's capacity derived. How the various modelling components can be combined to form a system model has also been described.

ACKNOWLEDGEMENT

This research was supported by the DTI as part of the LOng term Research into Civil Avionic Systems (LORCAS) programme and was funded by CARAD III.

REFERENCES

1. ARINC, 1994, "Integrated Modular Avionics Packaging and Interfaces", Aeronautical Radio, INC. (ARINC) Specification 650

2. ARINC, 1991, "Design Guidance for Integrated Modular Avionics", Aeronautical Radio, INC. (ARINC) Report 651

3. ARINC, 1997, "Avionics Application Software Standard Interface", Aeronautical Radio, INC. (ARINC) Specification 653

4. Baxter C.R., 1993, "The Airline Wish-List", International Avionics Conference, Integrated Avionics - How Far, How Fast?, Ramada Hotel, Heathrow, 1-2 December 1993, 1.1.1-1.1.5

5. Shlaer, S., and Mellor, J., 1992, "Object Lifecycles Modelling the World in States", Yourdon Press Computing Series, New Jersey, USA

6. SES/objectbench, 1996, "Technical Reference Manual", Release 2.3, SES, Inc., Austin, Tex

7. SES/workbench, 1996, "Sim - Simulation Language Reference Manual", Release 3.1, SES, Inc., Austin, Tex

8. Altia, 1996, "Altia Design Users Manual", Version 2.0, Altia, Inc., Colorado Springs

9. Little, J.D.C., 1961, "A Proof for the Queuing Formula: ", Operations Research, 9(3), 383-387

10. Penny, J.W.L., 1991, "A Review of the Control Technology Programme", International Avionics Conference, Advances in Systems Engineering for Civil and Military Avionics, Penta Hotel, Heathrow, 20-21 November 1991, 8.2.1-8.2.12

11. Goswami, K.K., Iyer, R.K., Young, L., "DEPEND: A Simulation-Based Environment for System Level Dependability Analysis", IEEE Transactions on Computers, vol. 46, no. 1, January 1997, 60-74.

TRAFFIC INFORMATION DISPLAY ENHANCING PILOT SITUATION AWARENESS: PARTI

K. Funabiki, National Aerospace Laboratory (NAL) Tokyo Japan,
S. Tenoort and F. Schick
Deutsches Zentrum fuer Luft- und Raumfahrt/ German Aerospace Center (DLR) Braunschweig Germany

ABSTRACT

A cockpit display for presenting air traffic information has been developed. The display, called "PARTI" (Pilot display of AiR Traffic Information), shows not only the positions of other aircraft but also their future trajectories and the locations of any estimated conflicts. The proposed display was evaluated by pilots participating in a series of part-task flight simulation experiments, and the pilots' situation awareness (SA) was evaluated in comparison with a display lacking a trajectory presentation. The functions of PARTI were mostly welcomed by the pilots, and were positively evaluated by them. It was also observed that pilot situation awareness was enhanced compared to the base-line display.

INTRODUCTION

It is anticipated that replacement of the current voice communication-based Air Traffic Management (ATM) system with a system in which communication is carried out over digital data-links may lead to a reduction in flight crew situation awareness of other air traffic[1]. In particular, the introduction of advanced ATM systems such as "Free Flight" which allow pilots a greater degree of freedom in flight planning than at present, requires enhanced traffic awareness for ensuring separation and avoiding conflicts. The Cockpit Display of Traffic Information (CDTI), which provides the flight crew with surveillance information relating to surrounding air traffic, has been studied for several years, particularly in coordination with the development of the Traffic Collision Avoidance System (TCAS), and has been implemented on the Navigation Display (ND) of a large number of transport aircraft. At present, several studies are being conducted in this area of research, considering recent developments in data-link technologies such as Automatic Dependent Surveillance-Broadcast (ADS-B) and the development of advanced ATM concepts[2].

The new display concept, called PARTI (Pilot display of AiR Traffic Information), differs from other proposed CDTIs in that it uses the flight plan information of other aircraft. Based on this strategic information, PARTI is expected to enhance flight crew awareness of other traffic, particularly for conflict avoidance, and to provide a more sophisticated aircraft collision warning system than TCAS. The use of flight plan information will be especially important for efficiency under four-dimensional (4D) ATM, under which most aircraft will be controlled precisely not only according to three-dimensional (3D) spatial constraints but also according to time constraints, and where they must follow predefined or negotiated 4D trajectories rather than simply maintaining a fixed altitude or heading. Although the transmission of flight plan information is not fully incorporated into the current ADS-B standards, future enhancements of ADS-B or a more sophisticated data-link system with a greater data transfer capability may provide for the transmission of flight plan data.

Situation awareness is a fundamental requirement for effective decision-making and performance. It can be described as the pilot's knowledge of the world around him and his place in it, and thus refers to "Knowing what's going on so you can figure out what to do". In this research, the definition due to Endsley[3] was adopted: "The perception of elements in the environment within a volume of space and time, the comprehension of their meaning, and the projection of their status in the near future." This definition highlights the three processing components of SA, which Endsley refers to as levels:

Level 1: perception of information

Level 2: comprehension of that information

Level 3: projection of its future implications

Figure 1 illustrates the hypothetical cognitive processes of a pilot engaged in a traffic avoidance task with and without flight plan information. In this hypothesis, the pilot's task is divided into three phases, namely "traffic situation interrogation", "conflict estimation", and "avoidance action". Each phase is related to one or two sub-processes of situation awareness or decision-making.

In the case of no flight plan information, if the pilot is well experienced and his workload is not very high, he can gain awareness of the air traffic situation by listening to radio communications between other

Copyright © 1999 The American Institute of Aeronautics and Astronautics Inc. All right reserved

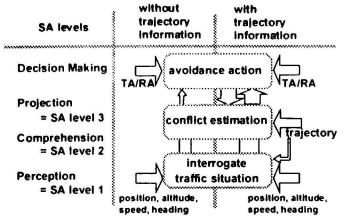


Figure 1. Hypothesised Cognitive Process of Conflict Avoidance

aircraft and air traffic control. Based on this "party-line information", he can estimate the positions and trajectories of other aircraft. When a traffic warning system announces the proximity of other aircraft and advises a conflict resolution action, the pilot takes avoiding action based on the advisory and his understanding of the traffic situation.

When the pilot is not well experienced, or if his workload is very high and thus the available resources for decision-making are low, the pilot will simply follow the advisory. In such cases, lack of situation awareness may cause pilot to make errors in responding to the warning and the advisory. On the other hand, if the flight plan information of other aircraft is explicitly presented, this is expected to enhance the cognitive processes of comprehension and projection. Using this information, the pilot can identify potential conflicts more easily and can easily understand the reason for any warning.

As part of the preliminary stages of this research, the study described in this paper focuses on the development and evaluation of a laboratory prototype of PARTI. Evaluation consists of determining the basic acceptability of the display and its concept and assessment from the point of view of situation awareness.

DISPLAY DESIGN

System Overview

The system configuration assumed for PARTI is shown in Figure 2. An aircraft's on-board traffic information computer receives traffic data including flight plans from other aircraft and from air traffic control (ATC) ground facilities. The aircraft's flight plan is also transmitted to other aircraft and to ATC facilities from its Flight Management System (FMS). Based on the traffic information, potential conflicts are identified and presented on the ND along with aural messages. A pilot can select and interrogate the traffic information by using a pointing device. He can also modify his own flight plan on the display and negotiate it with the controller by direct manipulation on the display using an interactive graphical user interface.

Intruders Classification

Detected aircraft are sorted into three categories: Proximate (PROX), Traffic Advisory (TA) and Resolution Advisory (RA). When a potential intruder's proximity decreases to within both horizontal and vertical thresholds and a computed Time to Critical Point (TCP) becomes less than 120 s, the intruder is classified as TA and a traffic advisory is issued.

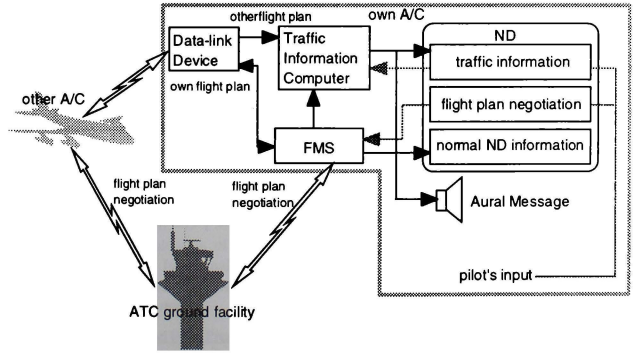


Figure 2. Assumed System Configuration

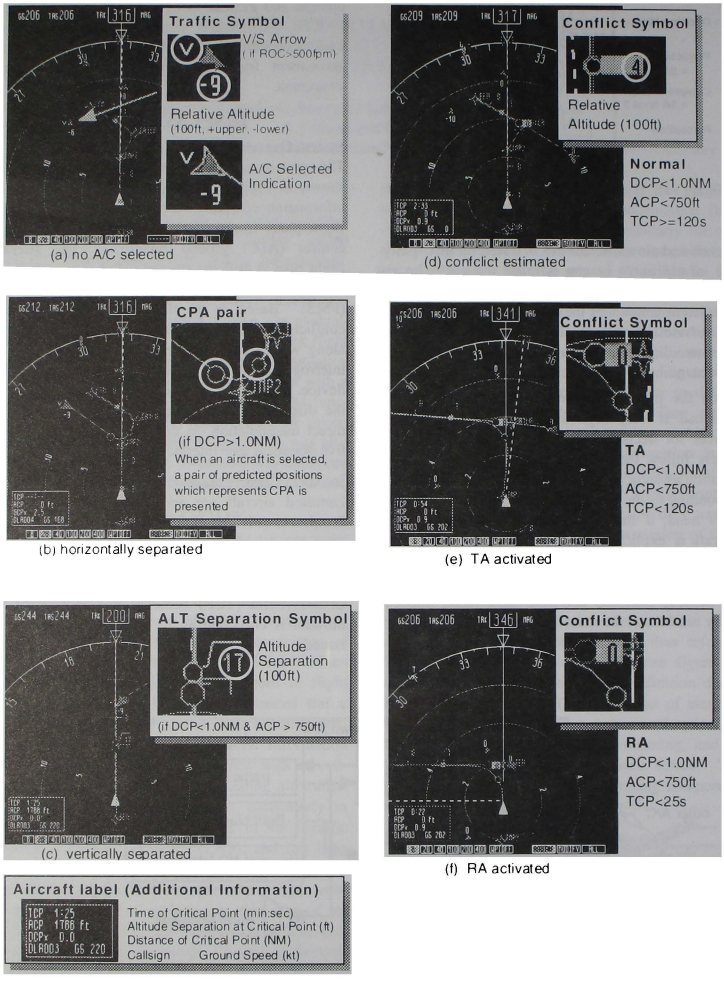


Figure 3. Display Symbology

When the TCP becomes smaller than 25 s, the intruder's category is upgraded to RA and a resolution advisory is issued. When the intruder is within both spatial proximity thresholds but is neither TA nor RA, it is classified as PROX.

The RA threshold time of 25 s was chosen to be equal to that of TCAS. The TA threshold time of 120 s was tentatively defined to allow for pilot decision-making and for trajectory negotiation between the pilot and ATC, and should be refined by further task analysis. Horizontal and vertical proximity thresholds were initially chosen as 1.0 NM and 750 ft respectively.

Symbology

The display symbology of PARTI is explained in Figure 3. A normal condition, where no conflict is estimated and no other aircraft is selected, is depicted in Figure 3(a). Relative altitude and vertical speed arrows are presented next to the traffic symbols.

When the pilot selects an aircraft on the display using a pointing device, its estimated future trajectory and a pair of critical positions indicated the portion of the trajectory in which minimum separation will occur are shown (Figure 3(b)). If the minimum horizontal separation is smaller than the horizontal threshold distance but the aircraft are vertically separated, an altitude separation is presented (Figure 3(c)). A small window showing TCP, ACP (Altitude of Critical Point), DCP (Distance to Critical Point), intruder callsign and ground speed is shown in the lower left corner of the display. When more than one aircraft is selected, this "additional information" corresponds to the most recently selected aircraft.

In that case that both predicted horizontal and vertical separations are less than their corresponding thresholds, a conflict symbol is shown (Figure 3(d)).

If a TA or RA is triggered, the intruder concerned is automatically selected and its trajectory displayed with a conflict symbol (Figures 3(e), (f)). In the case of a TA, the selected aircraft and its trajectory are colored orange; in the case of an RA, the color is red.

Some basic interactive flight plan modification functions were incorporated, borrowing from those of the Advanced Flight Management System (AFMS) developed at DLR[4]. A full 4D FMS capability was not necessary for the evaluation of the PARTI concept, and so only limited functionality was provided in order to demonstrate the procedures of traffic detection and conflict avoidance.

Assumed ATM System and Pilot Procedure

The design of the PARTI prototype assumed an ideal ATM system with the following characteristics:

-Aircraft follow 4D flight trajectories. Aircraft are guided by ATC not by heading, altitude or speed commands but by the proposal of new or modified 4D flight plans.

-Traffic information is shared between the air traffic controller and the pilot.

-The conflict resolution procedure is initiated by either the pilot, or by the controller, or by both parties.

-Flight plans are modified through trajectory negotiation between the controller and the pilot.

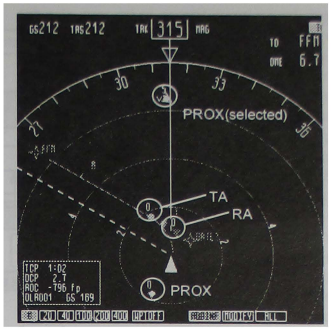


Figure 4. Experimental TCAS Display

The pilot's tasks for operation within this ATM system are divided into two categories: Normal condition and conflict avoidance. Under the normal condition, where no conflict is predicted, the pilot is expected to use the display to grasp the immediate traffic situation in order to be better prepared for unexpected changes of the situation or for possible future conflicts. Under the normal condition, this traffic situation awareness is not a "must know" requisite but "better to know" information. For conflict avoidance, the pilot should construct a new flight plan to resolve the conflict and start trajectory negotiation with ATC immediately following a TA. However, in the event of an RA, the pilot should take prompt avoidance action according to the advisory, either by manual control or through command-level autopilot modes such as vertical speed or altitude change modes.

It is further assumed that an on-board assistance system which supports the pilot in resolving conflicts will be required. Such a system should not only maximize the performance of the own flight and operation but should also coordinate with other aircraft to avoid subsequent conflicts which may be induced as a knock-on effect from the primary avoidance action. Although such a system may be necessary for real operations, it was not implemented for this phase of the study.

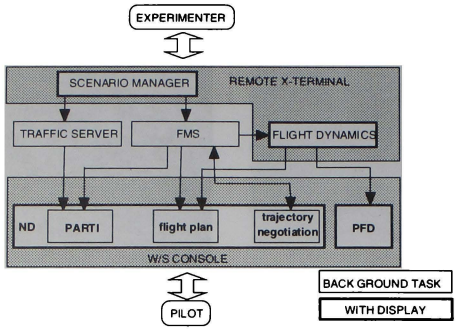


Figure 5 . Simulation Setup

Experimental TCAS Display

A display format and a conflict estimation algorithm based on current aircraft vectors were prepared for the purposes of comparison with the PARTI prototype. In this study, the linear estimation algorithm and associated display are referred to as the "TCAS mode" in comparison to the aforementioned "PARTI mode" which includes trajectory information. Figure 4 shows an example of the TCAS mode display. The symbology is largely the same as that of the PARTI mode except for the trajectory information. The classification of an intruder is shown as a small figure on its associated aircraft symbol, a white diamond, orange circle or red square corresponding to PROX, TA and RA classifications respectively.

Simulation Setup

Six processing tasks were integrated in a workstation: Flight dynamics, based on a non-linear mathematical model of a twin-engined medium size transport, with an autopilot; FMS; traffic server; scenario controller; Primary Flight Display (PFD); and ND. These tasks communicated with each other using shared memory and shared files. Traffic positions and conflict estimation were updated at 1 s intervals, and flight dynamics were updated every 60 ms. The PFD and ND tasks updated at approximately 0.1 s intervals. The pilot's PFD and ND displays were presented on the workstation's console. Another workstation functioning as an X-window terminal was connected via an Ethernet local area network, and was used for the experimenter's interface.

METHOD

Subjects

Eight pilots participated in the trials. All had the rank of a captain and their flying experience ranged from 1 800 to 20 000 flying hours. Six subjects had backgrounds in civil air transport, one was a research pilot, and one was a military and air ambulance pilot. Three subjects reported having had no experience with TCAS and four subjects reported that the aircraft they usually fly are not TCAS-equipped.

Experimental Design

Each subject's situation awareness was evaluated under two display conditions: TCAS and PARTI. Four pairs of similar scenarios (A and B) were devised based on the Frankfurt/Main airspace. The scenarios in each pair contained the same number and type of conflicts (see Table 1). In each scenario, the primary conflict was the encountered first and was designed such that the subject could estimate the probable conflict by assuming current flight vectors. Additional conflicts, following after the primary conflict, were termed secondary conflicts. The order of blocks and scenarios in the blocks were so arranged between the pilots to eliminate order effect.

Table 1. Scenario Attributes (H: Head-on , M: Horizontal merging, C: Catch-up)

Scenario	1A	2A	3A	4A	1B	2B	3B	4B
Number of A/C	5	5	5	6	6	5	5	6
Type of primary conflict	H	H	C	M	H	H	C	M
Type of secondary conflict	C	M	M	—	C	M	M	—

Procedure

Eight experimental sessions were held, with one pilot at a time. The evaluation consisted of two parts. A demonstration phase and an experimental phase. A questionnaire of acceptance was applied after each phase.

In the first part of the evaluation, subjects were first introduced to the concept of PARTI by three automatic demonstrations. The subjects were then asked to perform several tasks in three additional interactive demonstrations. These automatic and interactive demonstrations lasted as long as the subjects wished and until they were fully familiar with the concept of PARTI and with the operation of the Human Machine Interface (HMI). After this demonstration phase, the subjects were asked to complete a questionnaire concerning their acceptance of PARTI regarding aspects of the HMI and the operational concept.

The subjects were then introduced to the concept of situation awareness and to the different SA assessment methods which were to be used in the eight experimental scenarios which were to follow. The two experimenters demonstrated these methods using an example scenario to show the subjects what they were required to do. Thereafter, an additional scenario was presented to the subjects to allow them to practice the methods themselves. Further practice scenarios were performed as necessary.

Once the subjects were familiar with the methods, two blocks of measured runs were conducted, with SA assessment queries being administered after each run. Each block corresponded to one display condition (TCAS or PARTI modes) and contained four scenarios. Each scenario began approximately 130 s before the TCP of the first conflict, and continued until about 40 s before the TCP, whereupon it was frozen. The subject's task in this time was to grasp the traffic situation, and subjects were encouraged to select aircraft to obtain information in the additional information window. Once the scenario froze, the additional information window and conflict position indications were removed from the ND if present to avoid any bias in the subsequent SA queries, even if an aircraft was selected at the moment of the freeze. During the measured runs, no TA or RA aural warnings were issued. Furthermore, the subjects were not allowed to interact with the FMS, autopilot or flight controls. Throughout all measured runs and SA queries, the computer monitor together with the ND was recorded on videotape together with audio recordings.

Two objective methods for assessing situation awareness were applied in each simulation run. One is the Situation-Present Assessment Method (SPAM)[5]. For this, the ND remained visible and subjects were asked several questions relating to the positions of the own and other aircraft, any potential conflicts and their type, whether any avoidance action was necessary and if so what, etc. The order of the questions was varied between runs. During the query, subjects were asked to point the cursor at items about which they were asked. The primary dependent measure was the latency from a question being posed until it was answered, rather than the percentage of correctly answered questions. The rationale behind this method is that SA may involve simply knowing where in the environment to find a particular piece of information, rather than remembering what that piece of information is. The more time needed to respond to a question, the more the search for information is necessary and it is assumed that SA is lower at such times. The following questions were posed:

1 How many aircraft are in the scenario?

2 Which aircraft is the nearest to your own?

3 Is the nearest aircraft causing a potential conflict?

4 Are there other aircraft causing a potential conflict?

5 What type is the primary conflict?

6 Is an immediate avoidance action necessary, and which one?

7 Are there any further aircraft causing a potential conflict other than the primary conflict?

Questions 1 and 2 are related to level 1 SA, and questions 3–7 can be regarded as pertaining to level 2 and 3 SA. Note that primary conflicts did not necessarily involve the aircraft nearest to the own.

Once the SPAM query was completed, the ND was blanked and a method similar to the Situation Awareness Global Assessment Method (SAGAT)[6] was applied. Subjects received a schematic map of the airspace with waypoints and were asked to specify the positions of their own and other aircraft and to indicate the positions of any potential conflicts. For each aircraft recalled, subjects were asked to supply such information as heading, vertical speed, relative altitude and whether the aircraft was contributing to a potential conflict. Following all measured runs, a further questionnaire was administered which asked questions relating to the simulation and to the comparison of the two display conditions.

RESULTS

Acceptance

Two sets of questionnaires were presented to the pilots. One was after the demonstration phase to collect judgements on overall HMI aspects, and was on operational aspects of PARTI. Another questionnaire was presented after the experimental phase, and covered aspects of the simulation, and a comparison between PARTI and TCAS. The question-

naires consisted of total 36 questions. Results are summarised below, covering the different sections of the questionnaires individually. Some examples of the results are shown in Figure 6.

Overall, the layout of the display was positively accepted. The color-coding in general was judged as logical and comprehensive, and the relation of colors to traffic information as logical and unmistakable. Subjects reported that the meaning of the symbols was clear, they were clearly arranged on the display, and all text on the display was easy to read.

The general aspects of the HMI were also positively accepted, but two areas of criticism remained. Pilots quoted that the display sometimes contained too much information, and some difficulties during interaction were experienced when performing flight plan modifications and negotiations. The latter were attributable to the unfamiliar operation of the pointing device, and to occasionally slow response times of the workstation running the experiment.

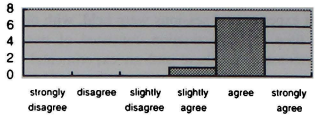
The participating pilots generally gave positive comments regarding improved traffic awareness, especially when comparing it to a conventional TCAS/CDTI after both blocks of simulation runs had been completed. These positive statements were supported by many of their answers given in the acceptance questionnaires.

Concerning the operational aspects covered by the questionnaires, all subjects agreed that PARTI assisted in conveying awareness of the surrounding traffic and in detecting and locating possible conflicts. As most of the subjects reported that they could easily recognize the intentions of other aircraft through their displayed planned trajectories, PARTI also supported the resolution of conflicts. All subjects considered that the display improved their situation awareness.

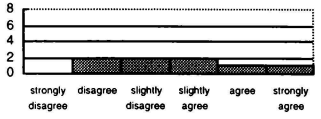
SPAM

Subjects' responses to the SPAM questions were analysed using the video recordings. Latency was measured as the time between the end of a question and the beginning of the response. Responses were scored as either correct or incorrect, except for responses related to conflict avoidance actions. The SPAM response data for all scenarios are summarised under the two display conditions in Table 2.

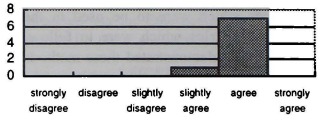
Generally the frequency of correct answers was higher under PARTI than under TCAS for all questions. Although no significant differences were found for the lowest level 1 SA (perception), the percentages of correct responses were higher under PARTI. Subjects reported the number of aircraft more correctly (PARTI 69.8%, TCAS 58.38%) and identified the nearest aircraft more frequently (PARTI 96.9%, TCAS 85.48%).



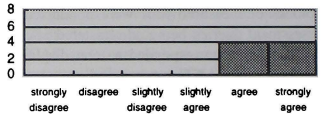
The meaning of the symbols used in the display was clear to me.



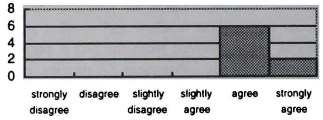
Sometimes the display contained too much information



I was clear about the intentions of other aircraft



PARTI was much more helpful to detect possible conflicts than TCAS.



The PARTI improved my situation awareness in comparison to TCAS.

Figure 6. Examples of Acceptance Questionnaires Results

Table 2. Results of SPAM Query (* significant at α= .05 level)

SA Level	Variable	PARTI Mean	PARTI SD	TCAS Mean	TCAS SD	SD
1	Number of a/c reported (% correct)	69.79	24.78	58.33	34.79	$F_{1,6}$= 1.788
1	Number of a/c reported (latency [s])	1.85	.63	3.41	2.31	$F_{1,6}$= 5.026
1	Nearest a/c (% correct)	96.88	8.84	85.42	23.88	$F_{1,6}$= 3.270
1	Nearest a/c (latency [s])	3.61	2.28	2.09	.79	$F_{1,6}$= 5.667
2	Potential conflict with nearest a/c (% correct)	88.54	16.02	63.54	36.71	$F_{1,6}$= 2.969
2	Potential conflict with nearest a/c (latency [s])	2.93	3.47	4.07	2.47	$F_{1,6}$=.829
2-3	Other conflicts than with nearest a/c (% correct)	100.00	0	46.88	20.86	$F_{1,6}$=45.63*
2-3	Other conflicts than with nearest a/c (latency [s])	6.34	6.55	6.09	5.32	$F_{1,6}$=.010
2	Type of primary conflict (% correct)	100.00	0	71.43	39.34	$F_{1,5}$=3.182
2	Type of primary conflict (latency [s])	2.04	1.62	1.14	.38	$F_{1,5}$= 1.803
2-3	Other conflicts than primary (% correct)	93.75	11.57	54.17	33.03	$F_{1,6}$=12.67*
2-3	Other conflicts than primary (latency [s])	7.28	6.05	6.92	6.38	$F_{1,6}$=.010
2-3	Avoidance action (latency [s])	3.29	2.38	2.27	2.13	$F_{1,6}$=.612

Significant differences in the responses were found for questions regarding higher levels of SA (level 2, comprehension, and level 3, projection). For the PARTI condition, subjects gave significantly more correct responses to question 4 (identification of conflicts with to aircraft other than the nearest) (PARTI 100%, TCAS 46.9%), and to question 7 (identification of conflicts other than the primary) (PARTI 93.8%, TCAS 54.2%). Although the responses to question 5 (identification of the type of the primary conflict) showed no significant differences between PARTI and TCAS, all conflict situations were correctly identified for the PARTI condition, as opposed to 71.48% responses correct for TCAS. The data were therefore analysed further to try to understand the particular types of conflict detected and the avoidance actions which the subjects would have chosen, although because the number of detected conflicts was much lower for the TCAS condition, no meaningful statistical analysis could be performed. Note that the subjects were not permitted to make any flight plan modifications but were asked to identify the type of conflict and to say what avoidance action they would have taken in response. Conflict situations were classified as one of three types: Head-on, catch-up and merging. The preferred avoidance actions were classified into the categories vertical path change (climb, descent or level-off), speed change, and turning. In two cases, subjects responded that they would not have undertaken any avoidance action but would have monitored the situation or informed ATC.

The results of this analysis are shown in Figure 7. It should be noted that some conflicts were reported as primary conflicts although they were in fact secondary conflicts. For example, the number of reported catch-up conflicts in the TCAS condition is higher than the possible number of primary conflicts shown in Table 1, because some primary head-on conflicts were neglected by the subjects and secondary catch-up conflicts were reported as primary conflicts. A considerably higher number of conflicts were reported as head-on under the PARTI condition (14) than with TCAS (4); similarly with merging conflicts (PARTI 7, TCAS 2). No such dramatic difference was shown between display conditions for catch-up conflicts (PARTI 9, TCAS 10).

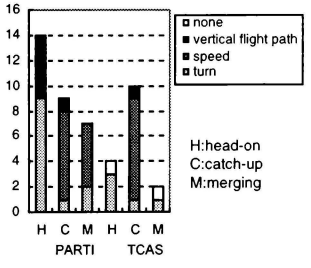


Figure 7. Type of Primary Conflicts Reported and Avoidance Action

Table 3. Results of the SAGAT Query (* significant at α= .05 level)

		PARTI		TCAS		
SA Level	Variable	Mean	SD	Mean	SD	F_{df}
1	Position error of own aircraft (NM)	3.22	.54	3.91	1.77	$F_{1,6}$= 1.491
1	Number of a/c correctly reported	4.06	.40	3.88	.80	$F_{1,6}$= .454
1	Absolute position error of a/c reported (NM)	3.41	.95	3.95	1.30	$F_{1,6}$= 1.127
1	Relative position error of a/c reported (NM)	3.54	.39	4.09	1.03	$F_{1,6}$= 2.706
1-2	Relative altitude of a/c reported (% correct)	22.92	13.64	35.94	16.14	$F_{1,6}$=4.163
1-2	Relative ground speed of a/c rep.(% correct)	31.20	16.22	35.36	17.42	$F_{1,6}$=.810
1-2	Vertical speed of a/c reported (% correct)	24.17	18.98	36.82	23.37	$F_{1,5}$=13.2*
2-3	Future flight path of a/c reported (% correct)	84.11	11.13	76.77	26.10	$F_{1,6}$=1.005
2-3	Number of conflicts actually present	1.75	.00	1.75	.00	
2-3	Number of conflicts reported	1.44	.37	.41	.42	$F_{1,6}$=83.77*
2-3	Primary conflicts reported (%)	87.50	13.36	37.50	37.80	$F_{1,6}$=16.00*
3	Secondary conflicts reported (%)	70.83	33.03	4.16	11.79	$F_{1,6}$=54.86*
2-3	Absolute position error of conflicts rep. (NM)	2.98	.91	2.32	.87	$F_{1,3}$=1.614
2-3	Relative position error of conflicts rep. (NM)	3.10	.90	3.18	.71	$F_{1,3}$=.013

Considering the relationship between type of avoidance action and reported type of conflict, no relationship with display condition could be detected since the low number of reported head-on and merging conflicts for the TCAS condition made such a comparison impossible. Overall, for catch-up and merging conflicts subjects preferred to make speed changes, whereas for head-on conflicts, initiating a turn was the procedure most frequently chosen. For this type of conflict, PARTI's display of information concerning the flight paths of other aircraft enabled the subjects also to consider changes to the vertical flight path. This was obviously possible because subjects had a better understanding of the intentions of the conflicting aircraft, for instance through indications of their top- and bottom-of-descent.

Analysis of the latency between query and response showed no significant differences between the PARTI and TCAS display conditions for any question. Furthermore, there were no clear relationships between latency and display condition, nor between latency and the level of SA probed by the question. Although there was no statistically significant difference in information search times between PARTI and TCAS, the number of correct responses was generally in favor of PARTI. Thus, it appears that longer search times were not significantly correlated with correctness of response.

SAGAT

Pilots' indications of each aircraft's location on the map were matched with the closet aircraft actually present at the time of the freeze, and the absolute (refer to waypoints) and relative (refer to own aircraft) position error in nautical miles (NM) was measured. The two measures for relative and absolute distance error of aircraft and conflict positions were taken into account to reveal if the two displays lead to different results in accuracy.

The total number of aircraft reported and the number of those reported correctly were counted. The latter accounted for instances when an aircraft was reported, but was not related to an aircraft position on the reference map used for evaluating the responses. Additional information pertaining to each aircraft (i.e. flight path, vertical speed, relative ground speed, and relative altitude) was scored as the number of correct responses, while missing responses were scored as incorrect. For each of the two display conditions the data of the scenarios are summarised in Table 3.

None of the questions related to level 1 SA showed significant differences between the PARTI and TCAS display condition, although responses under PARTI were generally better for these items. More correct answers were given for number of aircraft reported, and for relative and absolute positions of other aircraft. The responses for other aircraft's position were presumably better due to the presentation of their trajectories, which provided additional position information relative to existing waypoints.

Questions concerning additional information on reported aircraft, like relative altitude, relative ground speed, and vertical speed are considered to be related

to level 1 and level 2 SA. Results for these items were rather low, ranging from about 23 to 36 percent correct responses. These results reflect that subjects were fairly poor at reporting the dynamics of all aircraft in the scenarios, but tended to allocate their attention to the most relevant aircraft, i.e. the closet in time or distance. Nevertheless the percentages of correctly given additional information was generally higher under TCAS condition than under PARTI with a significant difference for the vertical speed (PARTI 24.2%, TCAS 36.8%).

When indicating the future flight path of other aircraft on the answer sheet, subjects performed better with PARTI (84% correct) than with TCAS (76% correct). Even though the difference was not significant, it may be taken as another indication for a positive effect of displaying trajectories.

Conflict detection performance was significantly better for PARTI (on average 82.3% of conflicts were detected) than for TCAS (23.4% conflicts detected), and more than twice as many primary conflicts were correctly reported for the PARTI condition (87.5%) than for TCAS (37.5%). The detection of secondary conflicts showed an even stronger difference (for PARTI, 70% correctly identified versus 4% for TCAS), which was also significant. This indicates strongly that it is very difficult to detect conflicts other than in the immediate future using the information provided by a conventional CDTI. However, for those conflicts which were identified, their absolute positions were reported with greater accuracy for the TCAS condition (a mean distance error of 2.32 NM) than for PARTI (2.98), which is reflected by an almost perfect 96.7% correct under TCAS compared to 74.2% for PARTI, though neither of these results were significant. This indicates that in the rather rare instance of detecting a potential secondary conflict with the TCAS/CDTI display, the subjects were almost always sure where this conflict would occur. However, it should be noted that three subjects were unable to indicate the position of conflicts under the TCAS condition in any scenario, which was reflected by a reduced number of degrees-of-freedom in the F-statistics for the conflict position data. Concerning the responses to the relative position of conflicts the picture was reversed, and subjects had slightly better results under PARTI than under TCAS, but again these differences were not significant.

CONCLUSIONS

PARTI, a prototype of CDTI using trajectory information was developed and evaluated in a series of part task flight simulations in which pilot's SA and acceptance were measured. The features of PARTI, especially the operational aspects were positively accepted, but few points of criticism about the HMI remained.

In comparison with a conventional TCS/CDTI the higher levels of pilot's SA (comprehension and projection) was enhanced with PARTI as the detection of potential conflicts was significantly improved. More than twice as much primary conflicts were recognised and only with PARTI secondary conflicts were reliably detected. No differences between PARTI and TCAS display existed at low level SA (perception).

REFERENCES

[1] Midkiff , A. H., & Hansman, R. J.: Identification of Important „Party Line" Information Elements and Implications for Situational Awareness in the Datalink Environment, Air Traffic Control Quarterly, 1(1), 5-30, 1993.

[2] Farley, T.C., Hansman, R. J., Endsley, M. R., Amonlirdviman, K., & Vigwant-Langlois, L.: The Effect of Shared Information on Pilot/Controller, 1998.

[3] Endsley, M. R.: *A methodology for the objective measurement of situation awareness.* AGARD Conference Proceedings: Situational Awareness in Aerospace Operations (AGARD-CP-478). Neuilly sur Seine, France: NATO-AGARD, 1/1-1/9, 1989.

[4] Adam, V., & Teegen, U.: PD/2 FINAL REPORT. Annex F Airborne Aspects of PD/2. EUROCONTROL DOC 97-70-13, Vol. 2. Brussels, Belgium: EUROCONTROL, 1998.

[5] Durso, F. T., Hackworth, C. A., Truitt, T. R., Crutchfield, J., Nikolic, D., & Manning, C. A.: *Situation Awareness as a Predictor of Performance for En Route Air Traffic Controllers.* Air Traffic Control Quarterly, 6(1), 1-20, 1998.

[6] Endsley, M. R:. *SAGAT: A methodology for the measurement of situation awareness.* (NOR DOC 87-83) Hawthorne, CA: Northrop Corporation, 1987.

[7] Midkiff , A. H., & Hansman, R. J.: Identification of Important „Party Line" Information Elements and Implications for Situational Awareness in the Datalink Environment, Air Traffic Control Quarterly, 1(1), 5-30, 1993.

[8] Dominguez, C.: Can SA be defined? In Vidulich, M. Dominguez, C., Vogel, E., & McMillan, G. (Eds.): *Situation Awareness: Papers and Annotated Bibliography (U). Interim Report AL/CF-TR-1994-0085, Armstrong Laboratory, Air Force Materiel Command, Wright-Patterson Air Force Base, OH.*, 1994.

REAL AVIONICS IN FLIGHT SIMULATORS FOR MILITARY AIRCRAFT

Dr. Yosef Gindin
Alexander Peleg
Elbit Systems Ltd.
Haifa, Israel

Abstract

In the modern aircraft world, in which avionics play a major role, an important issue for simulator is the accurate conformity to actual avionics. The system behavior in a modern aircraft is extensively affected by the software version installed in the on-board computer. The serious challenge to the trainer's architects is to provide the required conformity and to keep up with the frequent changes as the flight programs versions evolve. One of the ways to overcome this problem is to utilize the real avionics in the simulator. In this solution, the airborne software modifications seamlessly propagate to the simulator, which eliminates the need in the costly effort of modifying the simulator's software. Obviously this approach provides the ultimate solution for authenticity of the simulator. The need of the expensive process of the simulation testing is also eliminated.
This paper presents the concept of real avionics utilization in flight simulators for military aircraft, deals with challenges, which are inherent to this approach, and suggests the solutions.
The concept highlights:

- Installation of the real Pilot Vehicle Interface (PVI) devices in the simulator cockpit such as displays, panels and controls.
- Utilization of a real avionics computer with actual flying software.
- Accurate simulation of avionics sensors and other systems

This concept was developed and implemented in the Simulation and Integration Department of Elbit Systems Ltd. Haifa, Israel in several projects.

Role Of Avionics Simulation In Military Flight Simulators

Growing importance of Avionics in modern aircraft.

Today's military aircraft are equipped with huge amounts of avionics devices that assist pilots with handling the mission – devices that we have not seen in the prior generation of the aircraft. Better avionics allows better mission handling, sometimes even, against superior platforms. Today's sophisticated weapons can be operated only by means of computerized equipment. For this reason airforces from all over the world invest huge efforts to gain modern avionics and to upgrade old platforms. In fact, for the upgrade programs the modern avionics, is motto.
The modern cockpit is an integrated environment which includes the elements that we have not seen in the past, e.g. Head Up and Head Down displays, audio cues, head and eye tracking systems, on-board diagnostics and troubleshooting systems. The avionics today is a whole system designed with a common architectural approach and single operational concept. In this approach the avionics system is controlled by a powerful mission computer (may be more than one).
The key issue in modern avionics becomes the software and most of the development and testing effort is spent on its behalf.

How does modern avionics affect the training?

Efficient training requires authenticity of simulation of the complex mission variety. As avionics becomes more sophisticated so the simulation should keep up. The avionics becomes more interactive. It provides visual, audio and tangible cues that shall be accurate and realistic. Failure to mimic properly the cues reduces training efficiency and may even elaborate the negative habits.

Airborne software is frequently modified due to addition of new mission capabilities. The trainer should be capable to easily adapt to changes, otherwise it becomes obsolete in a short period of time.
Trainers' development, testing and maintenance efforts have always been an important issue but for the modern avionics it reaches a new scale of magnitude due to complexity of the avionics and due to the requirement for frequent adaptations.

Approaches to the Avionics Simulation

Here we try to review briefly the different approaches to the avionics simulation in trainers. Table on page 3 summarizes the key issues in each approach.

Approach 1. Modeling the avionics functionality on commercial computers.

This approach stipulates the software modeling of the mission tasks according to the Operational Specification and the Flight Manual. The models run on commercial computers and generate the avionics displays and audio using commercial hardware and software tools.
The mission environment (i.e. motion, sensors, weapons, radio, navigation sensors, etc.) can be modeled at a low level of similarity. No real I/O traffic shall be simulated – only the data exchange and logic of operation of the environment is pertinent to this approach.
An important advantage of this method is low hardware requirement and hardware maintenance costs of the commercial platforms.
The drawbacks are hereinafter:

- Large scale software development: the Operating Flight Program (OFP) shall be mostly re-written due to factors such as lack of availability of the OFP source code, incompatibility of computer platforms, languages, operating systems, tools, hardware and software interfaces.
- Difficulties in incorporation of real PVI devices such as displays, panels, indicators, etc. Operating such devices often require special I/O interfaces which are not available for the commercial computers. Normally the avionics PVI instruments are implemented with commercial replacement
- High software maintenance cost due to avionics changes. OFP modifications require changes in trainer's software.
- Difficulties in porting the software to other simulators. The trainer's software is designed to imitate the specific OFP behavior, which makes it irrelevant for a different aircraft.

Approach 2. OFP Rehosting on commercial computers

In this approach an attempt is made to adapt the OFP to run on a commercial computer. The rehosting effort involves the necessary adaptations to the OS and environment tools, development of the new I/O interfaces (e.g. display and audio) and adaptation of display formats (fonts, scales, symbols, etc.). Usually it requires restructuring of the software due to hardware platform incompatibility. Sometimes it requires rehosting of several program items running on different processors and implemented with different languages.
In spite of these difficulties and modification, still, large parts of the OFP can be reused. Therefore the future modifications of the OFP code can propagate to the trainer code thus reducing the software maintenance effort.
This approach also stipulates low hardware requirement and hardware maintenance costs of the commercial equipment.
As in the previous approach the mission environment can be modeled at a low level of similarity – only the data exchange and logic of operation of the environment is required.
The disadvantages of the method are the following:

- Significant software development efforts for OFP adaptation. It may be even more difficult than in modeling approach.
- High demands from software personnel who need to be familiar both with the OFP and the simulation code, as well as with the development environments.
- Difficulties in incorporation of real PVI devices (require real I/O).
- Significant maintenance cost due to avionics changes. Portions of code are modified manually.

Approach 3. OFP Emulation on commercial computers

The idea here is to run the real Avionics OFP (at binary level) on the commercial computer by emulating the OFP operational environment. In this case modeling of the environment shall be at high level of similarity – real I/O traffic shall be simulated.
This approach has a potential of low level maintenance effort and high level of similarity to the real avionics.
Unfortunately it is infeasible in the case of complex avionics due to the following factors:

- Complicated I/O environment of the mission computer shall be re-implemented in the new (commercial) platform. It includes non-standard format display generators, specialized communication channels (1553, ARINC, HOO9, Modems, etc) and multiple audio channels in dedicated formats.
- Heterogeneous operating environment of the avionics computers: multiple CPUs, multiple Languages (ADA, C, Assembler for Firmware, etc.). All these environments shall be emulated also.
- Very high computing power requirements of for the emulation of the non-native code.

In fact, an attempt to emulate the mission OFP leads to the rehosting option discussed above or to the mixture of both solutions.

However the OFP emulation can be considered for relatively simple systems such as for standalone instruments. For example the graphical S\W for avionics display can be emulated on a commercial computer.

Approach 4. Utilization of the Real Avionics

This approach stipulates the use of the real mission computers and the PVI devices while simulating the sensors and non-cockpit equipment through the actual avionics interfaces.

As in the previous case modeling of the environment shall be at a high level of similarity – real I/O traffic shall be simulated.

The simulation effort can be significantly reduced through reuse of the existing models available from other projects or from the subsystem providers. Many subsystems are standardized (e.g. SNU-84 for INS), some systems do not vary between the different aircraft (e.g. ADC) and some systems are commonly used in military projects (weapons).

This approach guarantees full authenticity and zero-effort simulator upgrade due to OFP changes.

The trainer development cost is very low due to maximum reuse of real avionics hardware and software.

The trainers designed with this method have potentially high manufacturing cost because the real avionics units are expensive. However comparing to the development cost in the approaches 1-3, this approach is cost effective. The cost can be somewhat reduced by utilizing the non-qualified for flight units.

Comparison of the Approaches

The next table compares efforts required for the described above approaches. Of course, specific project parameters can vastly affect the provided estimations. However this table briefly summarizes the above discussion and in our opinion it is representative.

Criteria for comparison	Approach1	Approach2	Approach3	Approach4
1. Similarity to real system operation	Problematic	High	High	Absolute
2. Efforts for upgrade due to OFP changes	Very High	High	Medium	Zero-effort
3. S\W development efforts	High, comparable to OFP S/W efforts creation	High	Huge, sometimes infeasible	Small
4. H\W development efforts	Small	Small	Medium	Medium
5. Manufacturing cost for replication	Medium	Medium	Medium	High

Conclusion

Following the above discussion we believe that for a small amount of simulators, which usually is the case, the real avionics approach is best suited.

Principle architecture of the simulator with real avionics

The block diagram of the avionics simulation core (ASC) for fighter aircraft simulator is shown in Figure 1.

The avionics PVI devices in the cockpit are real. Imitators with the same functionality and high degree of similarity can be used as well, depending on cost effectiveness. All avionics PVI devices are connected to the simulation computer and to the avionics mission computer through IO boards. The avionics mission computer is the real one with the actual OFP version.

The IO machine consists of various IO boards for interfacing with the mission computer as well as required adaptation and cabling accessories.

The avionics simulation computer through a standard bus interface (VME, PCI, etc.) accesses the IO boards. Many commercial IO boards are suitable, but in some cases the dedicated I/O should be developed.

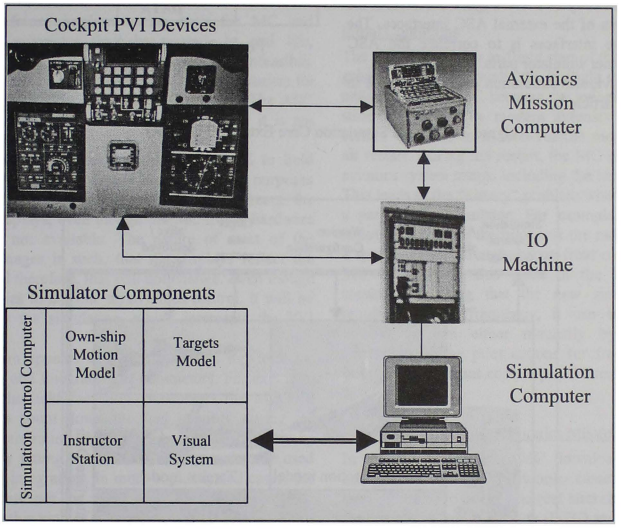


Figure 1. Avionics Simulation Core Block Diagram

The avionics simulation computer is a commercial brand. It runs the real-time simulation of the MC environment (navigation, weapon, communication systems, etc.). From our experience the majority of the cases the computing power of the modern PC is sufficient. However, more than one computer, if required, can be used in this architecture. Following are the main tasks of the avionics simulation computer:

- Simulation of MC environment (non-cockpit avionics subsystems)
- PVI devices model. Rather that connecting PVI units directly to the MC (as in the aircraft), we interconnect the units' data channels through the corresponding model (see Figure 2). This technique allows taking over the MC and the PVI control by the simulator. When the trainee controls the avionics, the PVI data channels are

short connected to the MC while the model monitors the traffic. During the playback or the instructor's "take over", the data stream is routed to the simulator.

- Drivers for IO machine control.
- Interface to other simulator components (see **Figure 3**).

The simulation computer can be used for aircraft motion and targets simulation as well. It does not affect the avionics simulation core architecture.

The simulation computer of the ASC is connected to the other simulator components through various data links such as shared memory or LAN. **Figure 3** describes the main data streams of the external ASC interfaces. The purpose of these interfaces is to correlate the ASC operation with other simulator parts.
We discuss the avionics simulation core architecture in the bibliography article [1].

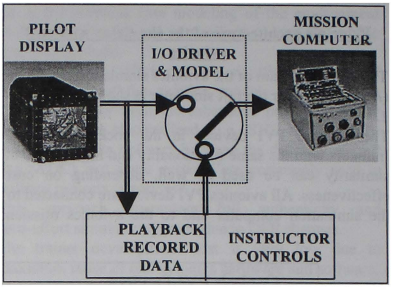
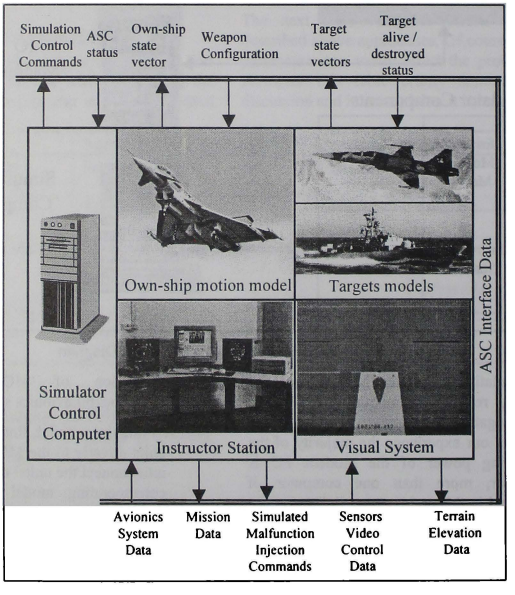


Figure 2. Cockpit PVI Devices Interface Implementation

Figure 3. Avionics Simulation Core External Interfaces



Implementation Problems And Suggested Solutions

Utilization of real avionics in the simulator presents several non-trivial problems dealing with cooperation of the simulation complex and the avionics mission computer. Here we intend to introduce the major issues and their suggested solution.

Time management

Simulation trials freeze. Upon freeze command the trainee expects all images in avionics displays to be frozen, until the resume command is issued. The continuity of all processes and the displays shall be maintained when the simulation resumes.

For ideal freeze implementation both the MC and simulation computer shall be paused. In real life, however, stopping the computers' clock is infeasible. Sometimes mission computers have such mechanism for debug purposes, that a simulator can utilize. The ASC commercial computer lacks such luxury, but it is not really required.

The simulation software should be designed to hold recent data during the Freeze State. For most purposes holding the simulated data is sufficient to freeze the avionics displays, even though the MC freeze hardware support is not available. The nature of most of the avionics images is such, that the displays reflect the sensors and therefore, that will not change. Even though the pilot tries to activate an avionics control, it will be blocked by the simulation, which intercepts the PVI devices.

The problem appears when the MC uses its clock for calculation and integration of parameters. For example, based on the held fuel flow parameters the MC will continue to count down the fuel quantity during the freeze. Overcoming this problem sometimes can be achieved by setting to zero the parameters that are used by MC for integration. In more sophisticated cases, the problem can not be resolved generally. From our experience this kind of problems do not seriously affect the training process.

Time of Day. The MC manages the time of day (TOD) function. It is used for the internal calculations and it is usually displayed to the pilot on one of the displays. The problem here is that the TOD shall correspond to a specific condition of the training mission, like light conditions of visual world, time of sensor models and the other gaming entities. The MC sets the initial system time from GPS (if available) or pilots can enter it manually. In the first case, initial time will be transferred to the MC through the GPS model. In the second case, the instructor can set the updated time at the beginning or during the exercise.

Data Synchronization. The data synchronization between the MC and simulated sensors should be handled properly by the sensor models. The models shall correct the computed data due to a synchronization event. The common source for such events may be 1 pulse per second (1PPS) signal from the GPS sync command or the sync command on the avionics bus. In case of an external synchronization source, the avionics simulation computer shall receive the signal and correct its clock accordingly. In situations where the source is the simulation, the problem does not occur.

Initial Conditions

The simulator allows in-flight start of the trial as well as ground starting. Normally the in-flight start is supported naturally by the MC because the avionics air restart is strongly required by modern avionics. MC maintains certain variables in its non-volatile memory for future air restart. During the restart, the MC recovers the last avionics system status, including the last pilot's setting. This leads to the "history" problem when the real MC is a part of the simulator. For example, if during the simulation session, the pilot sets the radio frequency to a specific value through the up-front control panel, the MC will recover this value in the next simulation session. Assuming that the new simulation session requires another frequency, it can be entered into avionics system either manually by pilot, or by simulation of the pilot actions for frequency settings through the up-front control panel interface (see Figure 2).

Simulation Mission vs. Avionics Mission

In modern systems the MC downloads the avionics mission data through the avionics cassette or from mass storage device (disk). In the real aircraft, the pilot takes the prepared mission cassette (DTC) and inserts it to the data transfer unit (DTU) before a flight. The MC reads this cassette according to the pilot commands. The DTC is prepared on the mission preparation station, which is a part of avionics ground support equipment. Obviously, the avionics mission data should be identical to the mission, defined in the training scenario by the instructor. For example, the weapon inventory defined in the cassette must be the same as the inventory of the weapon system model. Such correspondence may be achieved in several ways:

- Manual adaptation of the simulator mission data to the cassette data by the instructor. The practical way is to prepare a set of cassettes and predefined simulated missions beforehand.
- Use of the avionics mission preparation station as a part of simulator. In this case, the required data from a simulator can be exported from the avionics mission files or vice versa.
- Implement the data transfer unit model as one of the avionics models. In this case, the real DTU is not connected to the MC directly, but through the model. The model will read the data from the cassette, it will make the required settings in the avionics models and it will send this data to the MC. Such implementation is more complicated, but it carries two benefits. First, the changes in the avionics mission seamlessly propagate to the simulation. The second benefit is an ability to use the avionics debriefing data recorded by the MC during the flight. Such data includes weapon delivery information, deviation from the mission plan, activities log, etc. This data is valuable for the trainee activity evaluation.

Playback

In many cases a playback (or a replay) capability is required for the simulator. The following aspects are involved in this task:

- Recording of the simulated flight
- Storage of the recorded flight
- Loading of the recorded data for playback
- Replay of the recorded data from any point
- Freeze the replay at any point
- Continue of playback from the freeze point
- Switch to normal (interactive) simulation from the freeze point

Here we discuss the techniques of the playback implementation with real avionics mission computer in the simulator.

During the simulated flight the state vectors of own-ship and targets are recorded. All pilot actions due to the avionics system (buttons pressing, switches) are recorded as well. The data and the events are registered with the time tags.

The recorded data is loaded to the simulator during the playback initialization. The data is transferred to the MC through the corresponding models and I/O interfaces with respect of the recorded time tags. It grants the same behavior of the simulated avionics as in the original flight.

The playback shall always start from the beginning because only the full sequence of pilot actions brings the MC to a known state at a given point of time. This limitation of using the real MC presents a serious obstacle in a situation when the instructor starts a playback from some later point of time.

Our solution for this problem is to "forwind" the replay from the beginning in the "Fast" mode until the desired start point and to continue in the "Normal speed" playback mode from there.

In Fast mode the playback will skip the "dead" zones where no events were recorded, while the time intervals around the events are replayed in the real time. In other words, during the fast playback, the system "jumps" from event to event from the beginning until it reaches the desired start time. This method guarantees that the MC will be brought to a correct state before starting the playback at normal speed. **Figure 4** illustrates the training session with two events recorded before the start point Ts. First one is a pickle pressing at T1 and the second one is a missile release at T2. In this case the system jumps from the beginning to T1-Δ, replays it during the t1 time interval, then it seeks to T2-Δ, replays it during the t2, and eventually jumps to Ts. From Ts the playback resumes at normal speed until end of the recorded data.

The penalty of playing in the "Fast" mode is not significant since the sum of time periods when the events are played at a normal speed is small compared to a total flight time. We have measured 20:1 time compression ratios for a typical training session.

The events include all pilot actions (mode change, OSS pressing) as well as important changes in the simulated sub-system configuration (bomb release, simulated malfunction injection, Radar loss of tracked target, etc.). The list of the playback events with required time intervals should be defined during the system design and shall take into consideration the avionics system architecture and principle of operation. For the complex avionics system the list of events may include up to 200 events.

This approach is equally applicable for different types of desired system jumps such as "Reset to stored conditions" or "Rewind". In all the cases we will first reset the system to an initial state and then "forwind" it to a required point of time.

The Playback can be frozen and resumed at any point of time during the "Normal speed" mode.

Transfer from the playback to simulation can be done at any point by switching the MC data stream from the recorded data to the actual data stream (see **Figure 2**).

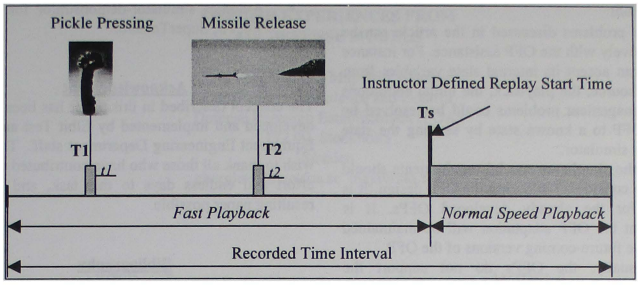


Figure 4. Fast Playback Illustration

Avionics Malfunctions Simulation

The imitation of avionics system malfunctions is strongly required for the simulator. The avionics sub-systems (weapon, navigation, communication, etc) malfunctions should be simulated in the corresponding models. The cockpit PVI device failures can be imitated by shutting down the devices or by insertion errors on MC interfaces that are routed through the simulator. The simulation of the internal MC malfunctions is very difficult with the real mission computer. In a training process, however, such failures are not critical because pilots do not have a recovery procedure for these cases.

Integration with Visual system

The Head Up Display (HUD) and Display and Sight Helmets (DASH) are important elements of the modern military avionics. Implementation of HUD and DASH depends on the simulator projection system. The real HUD is tuned optically to the infinitive distance. It means that the real HUD can be used only with collimated displays due to the fact that the superposition of projected images shall be focused to the same point. For non-collimated visual world displays, the HUD requires optical modification.

Another technique to overlay the HUD image is to combine the MC generated HUD signal and Out of the Window (OTW) image. Mixing of the high resolution OTW video signal with low resolution HUD video signal can produce such an overlay optically via an additional projector or electronically. In each case the mixing requires tuning of HUD image in terms of scale, location, color and intensity.

The mixing technology requires a raster format of the HUD video signal, however, usually the original HUD signal is generated in a stroke format. The HUD image in the raster format is available in many cases, from MC video recording channel. In lack of HUD raster signal (which is typical for older systems), the problem can be resolved by means of a commercial "stroke to raster" converter.

The real DASH implementation in the simulator raises the following issues:

- DASH optical tuning.
- Line of sight (LOS) tracking sensor accuracy.
- DASH image repeating for the instructor.

As in the HUD case, the real DASH shall be specially calibrated optically to the real distance from the pilot eyes to the OTW screen. However a single helmet is insufficient for the trainer because of different helmets sizes that are required for the pilots. The problem can be solved by fitting two or three different helmets for simulator needs. In the simulators with the collimated visual system, each pilot can fly with his own helmet.

The electromagnetic LOS sensors require an accurate magnetic mapping of the cockpit. The simulator cockpit can be mapped exactly as in the real aircraft with the same mapping equipment.

The DASH image monitoring for the instructor can be obtained with "stroke-to-raster" converter.

The aspects of real DASH integration are reviewed in the article [2].

OFP Adaptation

Many of the problems discussed in the article can be solved effectively with the OFP assistance. For instance if the OFP can accept its internal state variables from the external source, the playback, the initial conditions and time management problems could be resolved by setting the OFP to a known state by sending the state data from the simulator.

In this case the simulators special requirements should be taken into consideration during the OFP design. It is unpractical for the already developed OFPs. It is important that the OFP adaptation will be maintained along with the future-coming versions of the OFP.

However, usually, the OFPs do not support the simulator needs, which causes the simulator designers to look for solutions for problems that arise from working with real avionics.

Conclusions

After analyzing all the pros and contras of the concepts from technical, operational and maintenance and cost effectiveness aspects, we have decided to develop of the our new generation of simulators using real avionics.

We have successfully implemented the described approach in the numerous projects:

- Upgraded Mig21 avionics trainer
- Avionics simulation system for upgraded Mig21 full mission simulator
- Upgraded F4 simulator (in processing)
- Avionics simulator-demonstrator for upgraded F4, F5, SuperTocano.

Acknowledgment

The concept described in this paper has been designed, developed and implemented by Elbit Test and Support Equipment Engineering Department staff. The Authors wish to thank all those who have contributed their skills, effort and endless days to this task, and made this resulting paper possible.

Bibliography

1 Y.Gindin, E.Darom "Simulation tools within an aircraft upgrade program – a cost effective approach", AIAA-96-3522-CP, AIAA Flight Simulation Technologies Conference. A Collection of Technical Papers. pp.426-435

2 Jack D.Clevenger, John Mason Evans & Sutherland Computer Corporation "The NFG Battle Ground: Stimulation vs. Simulation for Effective NVG Training", MS&T-4/98, pp.4-9

SWEDISH EXPERIENCES FROM AIR DEFENSE MODELS

Erik Berglund and Maj-Britt Hansson
Department of Guidance and Control
Defense Research Establishment (FOA)
172 90 Stockholm
Sweden
eberglund@sto.foa.se maja@sto.foa.se

ABSTRACT

This paper presents experience from the use of computer models, mainly many-on-many models, for assessment of Swedish air defense weapon systems. The experiences are in most cases applicable to other fields of assessment than air defense. The main conclusions of this experience are that verification, validation and accreditation of computer models is not enough. The validity of the assessment is not dependent only on the details of the model and the validity of the model, but also on the ability to state the right questions and to transform these questions into MOEs and scenarios. Even simple questions can be answered totally misleading with a validated model if the MOEs and/or scenarios are chosen in a wrong way. Therefore it is very important that assessments are made in close collaboration between analysts and decision makers. The difficulties when using many-on-many models is the ability to understand the many dependencies. Therefore small transparent models are generally to be preferred to large, detailed models.

BACKGROUND

Computer models are often used to assess the effectiveness of air defense weapon systems. The Swedish Defense Research Establishment (FOA) has long and extensive experience of using such models. This paper presents lessons learned mainly from 10 years use of a many-on-many surface-to-air weapons model. The paper shows some problems occurring when using models, and the conclusions that can be drawn from that.[1]

Copyright © by Maj-Britt Hansson. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

MODELS

The models used for missile system assessment vary widely in complexity. The most commonly used models at FOA are:

- Kinematic one-on-one models, giving launch and intercept zones.
- Detailed 6-DoF dynamic models, giving miss distance against maneuvering targets using electronic countermeasures.
- Many-on-many models.

One-on-one models

One-on-one models treat one weapon system versus one target. The model is normally one sided, i.e. there is an a priori decision on which side will be the aggressor and which side will be the defender.

One-on-one missile models can be either kinematic or dynamic. The models can also be of two or three dimensions.

The required level of details in the different models varies greatly. There is, however, no guarantee that a detailed model will give a more correct answer than a model with lesser details. A more detailed model requires more data as inputs. If the data are uncertain, the model can give an answer that is less accurate that the answer provided by a less detailed model.

Often detailed data for the guidance and control system are not available, when assessing air defense systems. As the design of the real guidance and control system requires major efforts, there are no easy shortcuts available for the analyst, who is to model the missile system. This means that a detailed dynamic missile model with guidance and control designed with small effort can be more wrong for many purposes than a kinematic model would be.

One further advantage with kinematic models is that the reason for missile failure is readily available in the model, e.g. exceeded maximal time of flight. In the case of dynamic models there is no easy explanation available for the calculated miss distances. Bad performance of the missile can be true, but it can also be the effect of the insufficient control system design. [2]

As long as the target is not maneuvering dynamically or using electronic countermeasures, the firing zones from the dynamic model will be almost identical to the ones from the kinematic model.

As an analytical tool kinematical models should be used for most kinds of missile assessment. In those situations that involve dynamic evasive maneuvers and electronic countermeasures, dynamic models have to be used. However, great effort should be spent on validating the conclusions drawn from such simulations with respect to input and modeling uncertainties.

Many-on-many models

The current Swedish many-on-many model is called SILVIA. The development started in 1983 with discussions between Swedish Defense Agencies (the Army Air Defense School , the Defense Materiel Administration and the Defense Research Establishment) and the Swedish air defense industry (Bofors, Ericsson and SAAB) on how to develop new air defense models. Later the contractor Enator Telub joined the project. The main reason for the cooperation was a distrust of each others simulation tools. These discussions started the development of SILVIA, a family of models for air defense assessment. The idea was to provide Sweden with simulation tools for a long time. The original idea of SILVIA is close to the J-mass idea [3] but SILVIA is only handling air defense protection of one asset and the submodels are more tightly connected to each other than they seem to be in J-mass. It also seems that J-mass handles more command, control, communication and intelligence than SILVIA does and that the aggressor behavior is more detailed.

In 1987 the first production simulations started. Since then, the model has been used both by industry and by the Armed Forces for assessment of air defense from different point of views.

SILVIA has been used by Bofors in the development of the new surface-to-air missile RBS23 BAMSE [4] to such an extent that SILVIA is said to be the mother of BAMSE. SILVIA was also used by the Army to define the requirements on the system.

The use of SILVIA by the Army has been to compare systems and to assess new systems in the study process. In this process the Army, the Defense Materiel Administration and the Defense Research Establishment collaborate.

The Defense Research Establishment has used SILVIA in several studies, e.g. to assess the cruise missile threat.

The main purpose of SILVIA is technical assessment in a tactical environment. SILVIA addresses the many-on-many situation, i.e. many air defense units against many aircraft. The tactical aspect lays in the deployment of the air defense units, the rules of engagement used in the threat analyzer of the Command Control and Intelligence unit (C2I), and in the design of aggressor aircraft behavior.

The model is event driven and deterministic. The language is ADA and SILVIA runs under VAX/VMS, Unix and Windows/NT.

Air Defense

Normal size of the air defense units in SILVIA can be about 1-2 surveillance sensors and 3-9 firing units with local sensors, although the model can accommodate larger sized units. The model contains no communication, i.e. communication is modeled as either working perfectly or not working at all. The technical aspect of combat control is included in the model but the human factors are not modeled to any detail.

The components of the air defense units are:

- Surveillance sensors connected to Command Control & Intelligence Units, C2I
- C2I Units (one or more)
- Local Combat Control Units with or without their own surveillance sensors
- Firing Unit (Missile or Gun) with acquisition and tracking sensors
- Weapon (command to line of sight missiles, homing missiles, projectiles)

The sensors can be radar, IR or optical. These are used in different functions for surveillance, acquisition, and tracking. The most important surveillance sensor in the Swedish Air Defense systems is radar. The radar modeling is based on the radar equation with respect to atmospheric losses, rain, target radar cross section (RCS) and

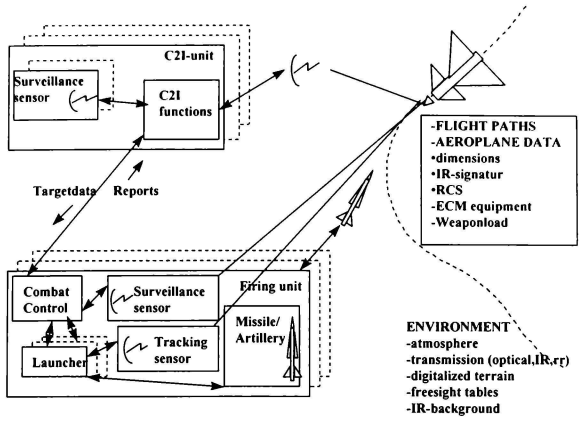


Figure 1. The scope of SILVIA.

countermeasures (ECM). Models of IR and optical sensors are generally less detailed.

The C2I Unit is controlling the combat of the air defense units and therefore has great importance to the events that will occur. It will gather information from the sensors, make the threat evaluation and cue the targets to the firing units.

The firing unit includes sensors and combat control functions.

Combat Control accepts the target designation and orders the firing unit to acquire the target if the firing unit has ammunition left. The acquisition and tracking of the target start and when the target is within range a missile is fired. The acquisition phase takes into account the time elapsed of reaction and searching.

The missile, according to its guidance principle, is either guided through ground based sensors or through its onboard seeker. The model takes into account that the missile can lose its track because of to long time of flight, ground impact, lost missile guidance and so on. If the missile termination is due to intercept of the target, the miss distance is calculated and the hit probability is calculated based on pre-calculated dispersion.

Aircraft which suffer hit probability in excess of 0.5 are considered killed and are removed from the rest of the simulation.

After the missile engagement, the combat restarts with surveillance. If no missiles remain at the launchers, reloading starts.

The missile models in SILVIA.

The missile model in SILVIA can either be described kinematical or dynamically. Both are three dimensional models.

The dynamic model means more accurate calculations of the missile movement and therefore requires more data in, for example more guidance, aerodynamic and motor data.

The different model levels have different approach to hit probability calculations and this requires different input. The kinematic model requires hit probability dispersion due to flight time and the dynamic model requires miss distance dispersion due to flight time to be added to the calculated miss distance.

The attacking aircraft

The flight paths of the aircraft are defined in advance. The only factor of the attacking aircraft that changes after the simulation starts is countermeasures on or off and that killed aircraft are taken out of the simulation.

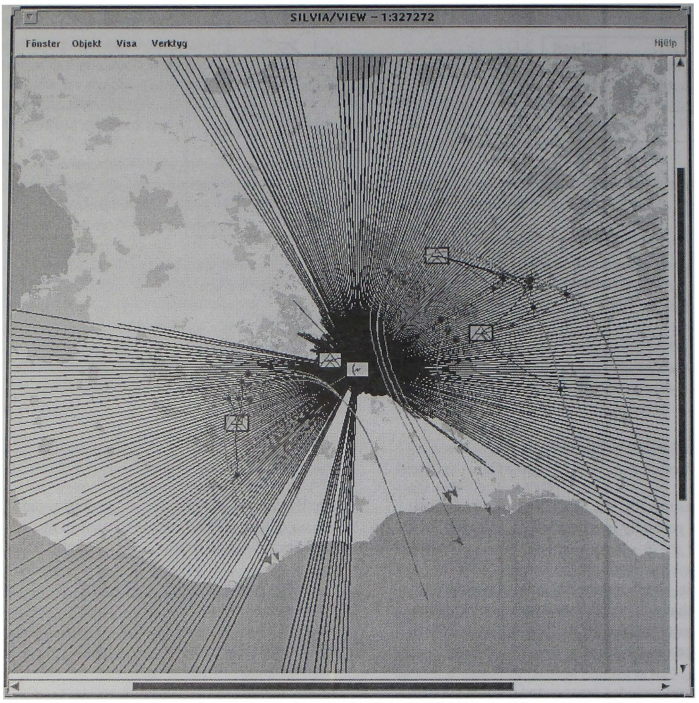


Figure 2. A moment picture from a simulation playback from SILVIA showing one air defense system with one surveillance sensor and 4 firing units protecting 2 air bases. The aggressor consists of 6 groups flying in on a low altitude (50m, 150 ft) in a tight formation. The straight lines show the free sight lines for the surveillance sensor.

The environment

The physical environment of the combat is terrain that affects the line of sight of the air defense units and also the deployment of air defense units and the low altitude behavior of the attacking aircraft. The terrain in SILVIA is a database of Sweden in 50 meter squares and line-of-sight tables measured from real deployment sites.

Other environment parameters are the surveillance sensors' capability to detect targets due to the curvature of the earth, rain, and light

VALIDATION OF SILVIA

When SILVIA was used in the study process to assess different air defense systems, the results from the simulations often were contrary to common sense. Often close to 100% of the attacking aircraft were shot down regardless of aggressor behavior or air defense system. Such a kill ratio is much higher than what war experience shows. This led to the suspicion that there was something wrong with the model. Therefore SILVIA was validated in 1994 through comparisons with a combined Air Force and Air Defense exercise, the FOCUS´94 exercise.[5]

The air defense system at the exercise was a mini system RBS90 [4] with one surveillance sensor Giraffe PS90 and one firing unit RBS90. RBS90 is a low altitude air defense system with round the clock capability, that in 1994 was just about to enter service in the Swedish Army. The target is tracked with an

IR/TV sensor and the missile rides a laser beam to the target.

The data recorded at the exercise was:

- surveillance data from the PS90 and from the firing unit surveillance radar PS91.
- tracking data from the firing unit
- digital signals from the firing unit
- tracking data from a reference tracking sensor, CIG790
- protocols from the surveillance sensor and from the firing unit
- video recordings
- position data recorded by the Swedish Air Force aircraft
- free sight tables measured at the deployment site.

No missile was fired at the exercise, so the result of the missiles were based on the video recordings, the protocols and the tracking errors from the firing unit.

Conclusions from the validation

- The result of this validation indicate no severe errors in SILVIA
- It also indicates that SILVIA is not as trigger happy as presumed. In some cases "reality" was even more trigger happy than SILVIA.
- "Reality" also shows very high kill ratios. At the exercise there was one attack with 6 aircraft and the single firing unit was able to fire at each one of them. That is exactly the kind of result that SILVIA is famous for and that caused the validation of SILVIA. In average when the defense could engage targets, it could shoot down 1 or 2 of the aircraft in a group of 4. This is a figure that also SILVIA usually gets.
- Overall the correspondence between SILVIA and "reality" was high. In some cases SILVIA and "reality" even lost the same missiles. When SILVIA did not fire and "reality" did, the missile was lost due to long flight times, i.e. SILVIA was right not to fire.
- SILVIA is validated and proved to be telling "the truth".

Although this validation showed that SILVIA was in some respect telling the truth, the problem with the discrepancies between SILVIA and war experiences still exists.

OBJECTIVE AND MEASURE OF EFFECTIVENESS

Using a computerized model for assessment demands more than knowing the buttons. To make a good assessment, the objective of the assessment must be clear and there must be a clear measure of how well the objective is met. [6] This Measure of Effectiveness (MOE) is not easy to define. [7] This is due to the fact that most of air defense system's effectiveness is not in the number of killed aircraft or the size of defended area, but in the effort the aggressor makes to avoid or kill the air defense system. This effectiveness is neither easily measured nor modeled.

One explanation to the credibility problem, where the kill ratio obtained by SILVIA and other models widely exceed wartime experience, is that the attacking aircraft and the defending anti-air units tend to have pursue different objectives. The aggressor aircraft tend to suppress the air defense units or stay out of their range, while the anti-air units tend to focus on killing aircraft that have come within range.

The difference in objectives emphasizes the need to use true two sided models for air defense assessment. Models that also handle the problem from the Air Force point of view would emphasize the air defense units abilities to avoid detection and to survive defense suppression rather than their ability to perform at the firing range.

This problem for all M&S is underlinded by the fact that wars no longer break down to small, easy to handle parts that can be aggregated to a next higher level. [8]

EXAMPLE OF RESULTS

The following examples will point out some of the difficulties when using a model like SILVIA.

Example 1:
The problems of choosing scenario and varying parameters is illustrated by an example where the Swedish version of the Hawk[4] surface-to-air missile was to be assessed. Two versions of the system were to be compared - the basic version with an old surveillance radar, RBS77 and an updated version with a much better radar, RBS87. Multiple scenarios with varied aggressor tactics were simulated, with and without the use of aggressor ECM. The percentage of aircraft killed by the two systems is shown in table 1.

	Without ECM	With ECM
RBS77	59 %	53 %
RBS87	58 %	42 %

Table 1. Percentage of aircraft killed.

One apparent, but totally wrong, conclusion is that the old Hawk system is better than the updated, especially in the ECM environment. A closer analysis, however, showed that the result was due to the fact that the Hawk system was deployed fairly close to the defended air base. The effect of the ECM was mainly

modeled as a reduction of radar range, but as most of the attacking aircraft flew close to the fire unit, the fire unit could still engage most of the aircraft. In the scenarios without ECM, the engagements were initiated at a long distance. The consequently long fly out times for the Hawk missiles often allowed the aircraft to escape, e.g. behind terrain obstacles. Another factor was that sometimes two missiles could be fired at short firing ranges for the same missile fly-out time elapsed for a single long range engagement. Hence, the correct conclusion was that one should not always use the maximum engagement range - which, however, in no way helped to answer the question on the cost effectiveness of the upgrade.

Example2:
The second example illustrates the importance of the scenario choice at a more detailed level. It also shows how the model details affect the result and how the choice of problem definition and the MOE affects the answer.

During the assessment of RBS23, the new Swedish low-medium altitude air defense system for air base defense, a question was asked if it would be best to have 2 firing units with 6 ready-to-fire missiles each or 3 firing units with 4 ready-to-fire missiles each. These configurations and a baseline scenario with only 2 firing units with 4 ready-to-fire missiles were simulated using a couple of different scenarios.

The first part of the assessment was 4 or 6 missiles for one firing unit. The example will illustrate a phenomenon due to deterministic modeling of reloading time and a scenario with exactly the same separation time between aircraft. Figure 3 illustrates the results from one particular case and figure 4 illustrates the effect of only a small change in reload time or a small change in time between groups of aircraft.

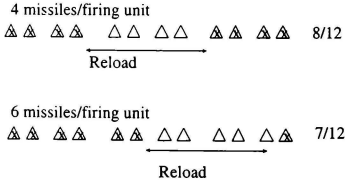


Figure 3. Illustration of killed aircraft from one firing unit.

In this example the result shows that it is actually better to have only 4 ready-to-fire missiles than to have 6, even though the reload time is not affected by having more missiles to reload. This is a strange result but depends on when the reloading is made. What the result actually says is that it is better to reload when there are no aircraft to fire at. In reality that conclusion is of no use, because the aircraft do not fly into the area at exact timing and neither is the reload time exact.

Figure 4 illustrates a scenario with the same configurations and a small change of the reload time. It could also illustrate the effect of attack groups in a closer formation.

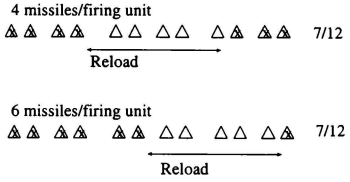


Figure 4. Illustration of killed aircraft. In this scenario the reload time is marginally longer than in figure 3.

The score in figure 4 is now even. With another small change the 6 missile alternative will be the better. This example illustrates that a very small change in the scenario details sometimes gives totally different conclusions.

The second part of this example illustrates the problem of defining the problem and finding the correct MOE.

The assessment concerned 2 or 3 firing units with 4 missiles. The conclusion was that 2 firing units was enough. All over again that result was due to the chosen scenarios, because all the scenarios were chosen so that the first firing unit had time to reload during the engagement of the second firing unit and the third firing unit had very little to do. This effect was also due to that the aircraft were flying at a rather high altitude (200 m, 600 ft) so the terrain had only a small effect. In reality the attack can come from either direction and therefore all directions need to be protected. This is especially essential when the attack is at a very low altitude where 2 firing units cannot get full coverage in all directions.

The conclusion that 3 firing units would be better than 3 at a given site was, however fairly obvious. The assessment that was essential in this case should not, however, had been to get the best result for at one site but to optimize for the protection of many assets.

As the attacking aircraft tend to spend much effort on avoiding or suppressing all sorts of anti-air systems, it

seems better to protect more assets even if that means a lower effect at each site. However, a many-on-many model like SILVIA is not able to address these issues.

VALIDITY

The above examples shows that although the model had been validated and that a multitude of scenarios were used, the analysis failed to provide any answer to the stated and fairly simple questions. The examples also show that the limit to analysis work is not given by the level of details in the model nor by the computer power, but by the analyst's ability to choose MOEs and scenarios that reflect the questions of interest.

The examples also raise many questions on what validity means for complicated many-on-many models. It is clearly not enough to build the model on validated technical data and to compare the behavior of simulations to reality or exercises.

EXPERIENCE FROM USING AIR DEFENSE MODELS

Scenario dependencies

The conclusions that can be drawn from a simulation are dependent on the chosen scenarios. With the wrong scenario the conclusions will also be wrong and that will mean that focus could be put on the wrong technical parameters.

When making an analysis, it is easy to focus on the simulated scenarios and forget the scenarios that are not simulated, because the answers are so obvious. Examples of these obvious scenarios are an attack in darkness against a daylight only system or an attack at high altitude against a short range system.

One reason for the high scores from SILVIA is this focus on scenarios that will produce some sort of result. We want to analyze situations where the air defense system is in action and therefore we choose such scenarios. The same phenomenon is also present in exercises. During an exercise we want some action to occur to train the operators. This is the reason why SILVIA and the FOCUS-exercise correspond so well.

In reality with true risks those simulated scenarios will not occur, as they would mean suicide for the attacker. To draw conclusions on technical parameter from scenarios that are so unlikely to occur will be totally misleading.
Most many-on-many models like SILVIA tend to exaggerate the importance of "firing range parameters" such as numbers of available missiles and reloading time. Combat experience, however, indicates that the "firing range parameters" are of little use in war. Models for air defense assessment need to be two sided and pay great attention to the problem of defense avoidance and suppression.

There is a great risk when scenarios are chosen in lieu with a clearly defined threat. Attention should be paid to the efforts required by the enemy to modify is tactics or equipment to defeat the system.

Model using

To be able to design the correct scenarios and to be able to draw relevant conclusions from a simulation with a many-on-many model, it is necessary to have a deep understanding of the air defense system. This means that the best way to make an assessment is to begin with small models and small problems. To start with a kinematic model is a good start and even paper and pencil can be highly useful. This knowledge of system behavior on a smaller scale is most valuable when analyzing on a larger scale. The main conclusion of this is that it is very important that assessment is made in collaboration between experienced analysts knowing how to use the model tool [9] and the decision maker who is the owner of the problem. A model is a tool for an analyst to make the problem more understandable.

One difficulty when using a model of SILVIA's complexity is the easiness to only see what you want to see. It is easy to explain strange results in the wrong way. The difficulty lies in that the phenomenon can depend on the model, the chosen scenario, the technical system or a combination of all these factors. It is easy to explain strange results with that there is something wrong with the model. That is because no one knows the total SILVIA. The model is too complex for one person.

To be able to make a good analysis the behavior of the systems must be correct. Otherwise the analysis will be flawed by the false assumptions.

SILVIA has been more useful as a model when the scenarios are small. The fact that the industry has used SILVIA with success indicates that this conclusion is correct. With large scenarios, too many parameters combine to a result that is too difficult to analyze. The dependencies are too many. This also points out that a model of more complexity than SILVIA will be even harder to use.

Experience shows that more scenarios should be used in each assessment and that it is essential that each scenario is varied with small differences. For example the Monte Carlo technique could be useful for the small variations.

CONCLUSIONS

The Swedish many-on-many air defense model SILVIA has been used extensively during more than a decade. SILVIA has been validated through comparison with exercises. The extensive use of SILVIA has also generated much general knowledge on weapon system assessment.

The Swedish experience of air defense analysis shows that the results of an analysis based on many-on-many models depends more on how the stated questions are transformed to MOEs and on the choice of scenarios than on the details of the model. Despite using validated models, answers, even to fairly simple questions, can be totally misleading. The main conclusion is therefore to underline the importance of experienced analysts working in close connection to the decision maker. Furthermore smaller, more transparent models are generally to be preferred to large, detailed models.

REFERENCES

1. Hansson M-B, FOA, Erfarenheter från användningen av värderingsmodellen SILVIA (Experience from using the assessment model SILVIA), FOA-R—97-00559-202—SE, (In Swedish)

2. Zarchan P, Tactical and Strategic Missile Guidance Second Edition, Volume 157, Progress in Astronautics and Aeronautics series, AIAA, New York , 1994

3. Bezdek W, Herr C S, McDonell Douglas Corporation, " Use of J-mass as a many-on-many engagement model" AIAA Flight Simulation Technologies Conference 1996

4. Jane's Land-based Air Defence, Eight edition, 1995-96

5. Svensson E, Rapport Validering SILVIA (Report from the validation of SILVIA), Enator 95120-45077A (In Swedish)

6. Naval Operations Analysis, Second Edition, Naval Institute Press, 1989

7. Berglund E, Hansson M-B, "Effektberäkningar (Calculations of effectiveness)" Swedish Journal of Military Technology 4-96, Militärtekniska föreningens tidsskrift 1996 (In Swedish)

8. Hill D C, The Strategic Assessment Center, "Modeling and simulation and the revolution in military affairs" AIAA Flight Simulation Technologies Conference 1996

9. Herrmann J, Aerodynamic Weapon Design Branch, "Air-to-air missile engagement analysis using the USAF Trajectory Analysis Program (TRAP)" AIAA Flight Simulation Technologies Conference 1996.

MISSILE MODEL FIDELITY AND INTEGRATION FOR AIR COMBAT SIMULATORS

Dr. Peter H Zipfel, AIAA Associate Fellow, Senior Aerospace Engr., USAF Research Lab, Eglin AFB, FL
Mr. Tom O'Grady, Aerospace Engineer, USAF Air Armament Center, Eglin AFB, FL

Abstract

Air combat simulators with their missiles are gaining increased importance as virtual prototyping is becoming a reality. After reviewing the dynamics of close-in air-to-air combat, we compare the fidelity of five and six degrees-of-freedom models of a generic short range missile. The 5 DoF model, adjusted for 6 DoF effects, is adequate for most situations. To speed up the insertion of missile prototypes into manned simulators, an automated conversion process from batch to real-time code is currently being tested.

Introduction

As all engineering endeavors, *air combat simulators* are benefiting greatly from increased computer throughput. With the latest Silicon Graphics processors, it has become possible to double the number of aircraft and missiles in simulated combat and to improve the fidelity of their models.

Air combat simulators are piloted flight simulators that engage multiple aircraft and missiles simultaneously. They support all phases of aircraft and missile development. During the requirement definition phase, aircraft maneuverability and missile fly-out performance are established; in the development phase, competing designs are evaluated; pre- and post-testing analysis is conducted; and, eventually, the airman is trained for combat.

Depending on the implementation, we distinguish between dome, work station, and virtual helmet simulators. At the current state-of-the-art, the dome simulators provide the highest situational awareness, but may be replaced by virtual helmet simulators in the future. A poor-man's solution is the tabletop workstation with the inherent limitations of a flat, screen display.

The software that drives the air combat simulators has a long history of development. The equations of motion of the vehicles are well understood. However, the fidelity of the models suffers under the limitations of the computer hardware. Although the aircraft are modeled in 6 DoF, the missiles currently, are simplified point-mass formulations in 3 DoF or in so-called pseudo 5 DoF.

As the developer relies more on prototyping by computer, it has become necessary to improve the fidelity of the missile models. Particularly for close-in combat engagements with their highly dynamic environments, the missile should be modeled with 6 DoF fidelity, or its simplified representation must at least be validated against it. Fortunately, the continued increase in computing power will eventually allow full 6 DoF representation of aircraft *and* missile models in air combat simulators.

To shorten the time of the design and evaluation cycles, an efficient process must be in place to convert from the design simulation to the embedded missile simulation. The 6 DoF model that is used to define the performance of the design does not have to be run in real-time – it is more likely executed in the batch mode. This *batch simulation*, which is a complete program, must be converted into a *subroutine* so that it can be embedded into the air combat simulator and must be capable of real-time execution.

The **objectives** of this paper are to (i) review standard maneuvers as they affect modeling choices, (ii) summarize our experience with missile model fidelity, comparing 5 DoF with 6 DoF implementations, and (iii) describe the conversion process of batch simulations to real-time code embedded in air combat simulators.

Background

It has become traditional to describe most air-to-air missile engagements in terms of the launch range, giving rise to the general engagement categories of Beyond Visual Range (BVR), and Within Visual Range (WVR). These are qualitative categories, because there is no specific launch range that can be identified as the sole criterion for visual detection of the opponent. Many variables influence the visual detection threshold, from atmospheric conditions to the choice of aircraft color schemes. It is possible for one combatant to be within visual

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

range while his opponent is not. The end result is some overlap between BVR and WVR conditions, due to variations in the visual detection range. Close-in combat (CIC) encounters are a special subset of WVR engagements. All CIC engagements are WVR by nature, but not all WVR engagements qualify as CIC.

The defining characteristic of CIC engagements is that the opponents are maneuvering at high load factors almost continuously. This high-g maneuvering results in a progressive loss of both speed and altitude, and flight paths that appear to be descending spirals. This is essentially the "dogfight" model, reduced to two combatants maneuvering one-on-one.

The feasibility of using air-to-air missiles in CIC conditions is a rather recent innovation. For over forty years, the view of air-to-air weaponry has presumed the existence of three complementary systems with overlapping, concentric coverage. Long or medium range air-to-air missiles (MRAAMs) were tasked with all BVR engagements. Since the only means of target detection in BVR conditions was the aircraft fire control radar, MRAAMs usually relied upon some type of compatible radar frequency seeker for guidance. Large warheads were required to compensate for the inaccuracy of earlier generation radar seekers, and MRAAMs tended to be large vehicles with relatively sluggish flight characteristics. Short range air-to-air missiles (SRAAMs) were preferred for the outer portion of the WVR engagement envelope. The typical SRAAM used an infrared seeker, since acquisition ranges were usually consistent with visual detection, and they were much less costly to build than contemporary radar seeker designs. As a rule, SRAAMs were lighter and more agile than MRAAMs, because of lighter warhead and motors, made possible by smaller miss distances and fly-out ranges. The gun remained the weapon of choice in CIC conditions, primarily because there was no practical alternative.

Today, air-to-air missile technology has progressed to a point where *bona fide* options are available for future system development. It is now realistic to design an air-to-air missile that is capable of very creditable performance in CIC conditions. A single, range insensitive missile can be effective across the entire spectrum of detectable target conditions. Development of such a system could be rather costly, and it would be very beneficial if the effort could be validated using proven analytical techniques. Unfortunately, while many analytical methods are available from past missile development programs, none have been shown to be truly satisfactory in CIC conditions. Analyses of CIC conditions have generally concentrated on aircraft performance because of the importance of flight quality for effective aerial gunnery. The usual assumption was that the CIC problem had been solved if the shooter could point his aircraft at the target, and hold that geometry long enough for his gunfire to take effect.

To investigate the performance of a *missile* prior to entering a manned simulator, we introduce typical launch scenarios that lend themselves to computer analysis. We concentrate on CIC engagements as the most challenging geometry.

Circle Fights

The defining characteristic of CIC engagements is that the opponents are maneuvering at high load factors almost continuously. Historically, the most common form has been a hard lateral turn in a steeply banked attitude. Although this maneuver is based on the presumption that a body-fixed, forward firing gun is the primary air-to-air weapon, the same tactics model has been used in analyses of short range missile engagements up to the present time. Viewed from above, the resulting flight paths of the opposing aircraft resemble a series of circles, leading to the terminology of *circle fight*. If viewed from the side, each successive circle fight would appear to be somewhat lower than the previous one, owing to the tendency of aircraft to lose altitude when in a steep bank. Overall, the flight paths of all the participants in a CIC engagement are descending spirals, as illustrated in **Figure 1.** High-g maneuvering usually results in a progressive loss of speed, as well as altitude, as each combatant attempts to reduce his turning radius . **Figure 1** is essentially the dogfight model, reduced to two combatants maneuvering

There is no single description of a CIC engagement that can be called 'typical'. It consists of a sequence of circle fights, but the number of individual fights in this sequence cannot be predicted in any general way. Furthermore, there are several distinctive types of circle fights that may occur in any order, and with any degree of repetition. This is a matter of pilot preference; and pilots on both sides have a say in the final outcome. Training doctrine cannot, therefore, make the analytical situation deterministic. In the face of all these ambiguities,

the customary approach to CIC analysis has been to concentrate on the individual circle fights which are the building blocks for an engagement, and to make a qualitative estimate of how individual circle fight results might translate into overall weapon effectiveness.

There are two fundamental types of circle fight that are important to CIC analysis. These are

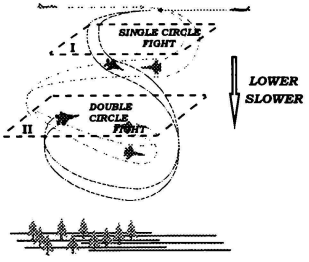


Figure 1 A CIC engagement as a sequence of circle fights

the single circle and double circle models. In the single circle fight model, both opponents fly in a constant altitude circular flight path about a common center of rotation. There are two essential subsets of the single circle model, depending on whether the combatants are flying toward each other, or are chasing each other. In the U.S., the chase version of the single circle fight is known as a *Lufbery circle*. The head-on, single circle fight, however, is considered the more important baseline engagement. In the double circle fight model, the opponents fly non-concentric circular flight paths that are tangent at a single point. The double circle fight model assumes that the opponents are turning the same way, either clockwise or counterclockwise. Were the opponents to turn in opposite directions, the scenario would be the default single circle fight model.

Prior to the introduction of air-to-air missiles, these circle fight models could be assessed relatively easily. The outcome depended on who could first point the nose of his aircraft at his opponent, and keep it pointed there long enough to inflict significant damage with his gun. The deciding factor was the aircraft performance, especially in the areas of maneuverability and durability. A design challenge arises from these conflicting requirements, since most design practices that would enable an aircraft to withstand punishment would also tend to increase its weight, thereby reducing maneuverability. Finding the proper balance between these divergent goals, derived from the characteristics of available air-to-air armament, shaped the course of fighter development for a number of years.

The introduction of air-to-air missiles had a major influence on this situation. A revolutionary concept of air combat had come into being wherein circle fights were irrelevant. It was no longer essential for fighter aircraft design to follow the traditional trend based on the body-fixed gun as the principal air-to-air weapon. There was, instead, a strong indication that air-to-air missiles, and the aircraft to carry them, should be designed as integrated systems. For a time, the use of circle fights as a criterion for weapon and aircraft performance faded into the background. Re-emergence of circle fights as a measure of performance in the CIC arena is a recent development, made possible primarily by advancements in missile and aircraft control technology. These advancements have increased attainable launch and flight envelopes by a significant amount.

For this study, the application of circle fight models to the analysis of air-to-air missiles will be emphasized. This emphasis is believed warranted by the different levels of analytical capability which exist between missiles and aircraft. There is no current analytical technique that is known to give credible system effectiveness estimates for air-to-air missiles in CIC engagements. Several promising methodologies are in development, but it will be years before the adequacy of these can be determined. At present, the most that can be achieved is a standard determination of missile performance in circle fight conditions. This is not without value, since it provides a quantitative way of expressing the performance requirements of missile systems, and of specifying the performance level to be demonstrated during missile system development. What is lacking is a proven way of measuring one missile system against another, to determine which design is most effective overall, and to give some feedback to the requirements definition process.

The format most often seen in the U.S. is the Launch Acceptable Region (LAR) graphic.

There are five distinct LAR graphics but those most relevant to circle fights are given the designations LAR-3, LAR-4, and LAR-5, representing the single, double, and Lufbery circles respectively. The procedure for construction of a LAR graphic can best be described through use of an example.

Figure 2 illustrates the major features of the LAR-3 scenario. The intention is to simulate a

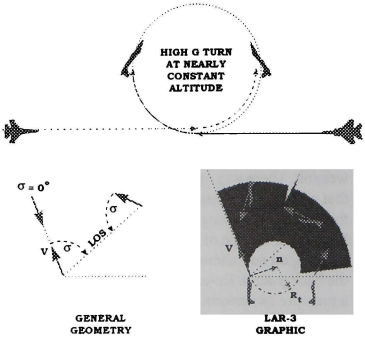


Figure 2 LAR-3 Single circle fight characteristics

default single circle fight, consisting of a constant altitude encounter with participants flying toward each other on a circular flight path at equal speeds. If a head-on geometry is achieved, which would indicate that the starting positions were further apart than the turning diameter determined by the maneuver load factor, the opponents will fly straight and level towards one another. At short engagement ranges, the result is a circle fight; at longer engagement ranges, most of the encounter is a simple head-on attack. The longer ranges are of little interest since LAR-3 is intended to apply primarily to CIC conditions. The general geometry of a single circle fight can be expressed in terms of the relationship between the velocity vectors of the shooter and the target, as shown in the lower left panel of **Figure 2**.

The *slew angle*, σ, is the angular displacement of the line of sight (LOS) to the target velocity vector at launch. If, for any selected value of σ, the target velocity vector makes the same angle to the LOS, the result will describe a single circle fight. In a high-g horizontal turn, both aircraft will be sharply banked, and the value of σ will be approximately equal to the sum of the shooter angle of attack, α, and the look angle λ (the angle between the nose of the shooter and the LOS)

$$\sigma = \lambda + \alpha$$

The graphic presentation of LAR-3 envelopes follow the format illustrated in the lower right panel of Figure 2. The slew angle σ is swept incrementally from zero to the largest off-boresight capability of the missile seeker. At each value of σ, the launch range is varied incrementally to identify those initial range conditions which result in hits. The shaded area shown in **Figure 2** is the area of successful launch conditions relative to the shooter, termed the launch acceptable region.

Two features of the LAR-3 graphic should be noted. First, the full capability of the missile is not normally illustrated. As mentioned earlier, the LAR-3 scenario is intended to depict CIC performance, and it is customary to truncate the envelope at a launch range of approximately six kilometers. The full performance envelope of a missile could extend to launch ranges of many nautical miles, but these longer ranges are of little interest.

A second feature of the LAR-3 graphic is that launch ranges inside the flight circle of the shooter, defined by the turning radius R_t, are not usually considered. The reason for this is that launch ranges within the flight circle result in a crossover between the flight paths of the shooter and target, and both opponents must turn through at least 180° before a true single circle geometry can be established. The situation is physically realistic, but it does not fit the "choreography" selected for LAR-3.

We will use the LAR-3 engagement to study the fidelity issues associated with close-in aerial combat. Although, there are several other types of engagements – the twin circle fights and the British 'combat circle' diagrams – the LAR-3 is representative of the stressing engagements of typical CIC encounters.

Generic Short Range Missile

To carry out our performance studies, we employ a generic short range missile, which was given the designation SR1S. **Figure 3** depicts the lay-out and the major subsystems. A high fineness ratio body-tail design was chosen as the baseline. A missile diameter of 6 inches was selected to lie in the center of the design space identified by the

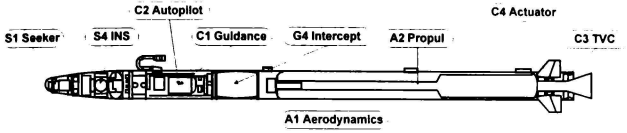


Figure 3 Generic Air-to-Air Missile SR1S with Subsystems Identifiers

physical design constraints. Forebody subsystem packaging includes an imaging infrared seeker , related electronics, an inertial navigation system (INS), a seeker cooling bottle, an active optical target detector, and a nominal 20 pound warhead with an electronic safe and arm system. The forebody shape is a 3:1 ogive, blunted by a hemispherical IR dome with a radius of 1.5 inch. Overall length is 116 inch.

Simulation Models

The missile concept SR1S is modeled in the CADAC[1] environment. It consists of 5 DoF and 6 DoF fly-out versions. The *target* is modeled as a point mass vehicle with its attitude aligned in the maneuver plane. The maneuvers are defined prior to missile launch.

employing tabulated trimmed aerodynamic data. The two rotational degrees of freedom are pitch and yaw (skid-to-turn). They are modeled by linearized differential equations that describe the attitude dynamics of the controlled airframe. In this case, Euler's equations are *not* solved.

The 6 DoF version is a *full* 6 DoF simulation solving the three translational degrees of freedom with Newton's equations and the three rotational degrees of freedom with Euler's equations.

During the missile design phase, CADAC is executed in its *batch mode.* Its many post-processing tools support the performance evaluation of the missile concepts. The same missile simulation can be stripped of all unnecessary subroutines using the *converter program* that generates a self-contained subroutine suitable for *real-time* execution. Thus

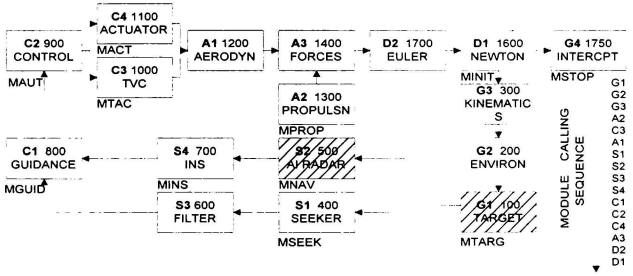


Figure 4 SR1S 6 DoF Simulation

The 5 DoF CADAC versions are so-called *pseudo* 5 DoF implementations. Three translational degrees of freedom are modeled by nonlinear differential equations (Newton's equations)

the traceability of the missile simulation from batch processing to MIL(man-in-the-loop) simulation execution is maintained.

The 6 DoF architecture of the batch simulation is shown in **Figure 4**. Each module represents a major subsystem with its closely controlled interfaces. The first two characters in the blocks identify the module, followed by the first C array location reserved for this module.

[1] CADAC (Computer Aided Design of Aerospace Concepts) CD-ROM, Version 3.0, Air Force Research Laboratory, Eglin AFB, FL. Distribution unlimited.

For all modules, 100 adjacent locations are reserved. Some of the modules are controlled by *mode switches*. They are shown beneath the module block. The *calling sequence* is important and must be maintained for the simulation. For real-time applications, the two blocks with hashed lines, *Target* and *AI Radar*, are deleted and are replaced by inputs from the flight simulator.

The 5 DoF simulation, presented in **Figure 5**, was built from the 6 DoF by simplifying the aerodynamics, and replacing Euler's equations by fairly good approximation of a radar scope, but cannot reproduce the visual field of view of a human pilot. Situation awareness is lost under CIC conditions, and with it, the realistic pilot response which was the primary goal of the exercise.

Time-of-flight of the missile and the endgame determination of hit or miss are the very foundations of situation awareness. Pilot actions form an unbroken chain of events, based on what the pilot perceives the situation to be. If

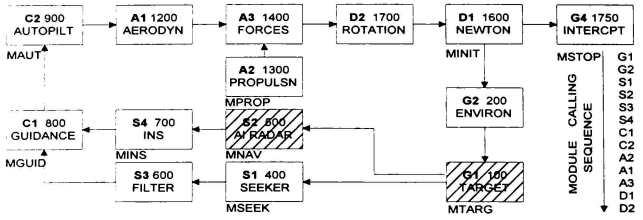


Figure 5 SR1S 5 DoF Simulation

the response of the attitude autopilots. Notice also, that the actuator and the thrust vector control (TVC) modules are no longer required.

Fidelity Comparison: 6 DoF vs 5 DoF

The principal requirement for any missile simulation in an engagement model is that it should accurately reproduce that sequence of events which affect the pilot's **situation awareness**. The major elements of this sequence are the *missile time-of-flight*, the outcome of the firing: *hit or miss*, and the time required to assess the *damage* inflicted by any hits. This last item is closely related to the missile lethality model, but consideration of lethality is beyond the scope of this study.

The importance of situation awareness cannot be overstated. The basic rationale for a MIL simulation is to capture the reactions of the pilot, and factor them into a better assessment of weapon effectiveness. The reactions of the pilot will be altered if his perception of the overall combat situation is contaminated by simulation errors, and most benefits of a MIL simulation environment will be wasted. This is the principal reason why workstation MIL simulations have been useful in beyond visual range (BVR) engagements, but not in CIC battles. The tabletop monitor can provide a the missile simulation time-of-flight is in error by a half-second, the actions of the pilot during that half-second are ambiguous relative to an actual launch. Initial conditions for any subsequent launch have been changed irreversibly, and system effectiveness conclusions may have been reduced to speculation. The same can be said of the hit assessment, since the actions of a pilot who perceives a miss will normally be quite different from one who sees a hit.

The fidelity of the missile model must take into account these requirements. From the standpoint of the simulation facility, the missile code should be as compact as possible and executable at the same time steps as the aircraft. That is, low fidelity models would simplify the integration task. However, the simplifications should not jeopardize the pilot's situation awareness. Since CIC is the most demanding engagement, we look into the adequacy of 5 DoF models to represent the missile fly-out in LAR-3 type envelopes.

Time-of-Flight Comparison

Time-of-flight is a critical parameter in determining the adequacy of any missile

simulation. A comparison of a fly-out trajectory in the middle of the envelope is shown in **Figure 6**.

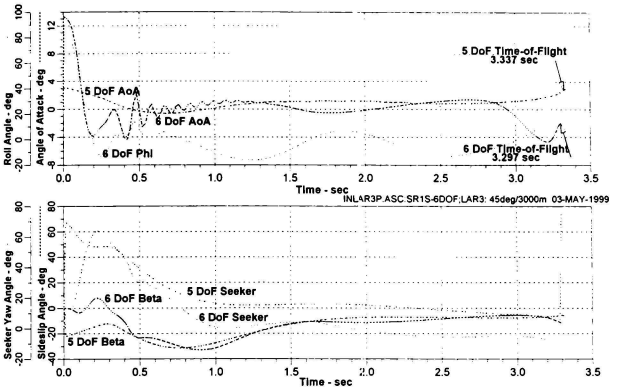


Figure 6 Comparison of 6 DoF and 5 DoF trajectories. LAR-3: R = 3000m, $\lambda = 45^0$

It is instructive to follow the incidence angles from launch to intercept for the two models. The 6 DoF simulation replicates accurately the initial roll angle, imparted to the missile by the shooter aircraft executing a 7.5 g maneuver at an angle of attack of 13 deg. The missile rolls out to a 'hooks up' attitude and develops sideslip angle for the lateral intercept. On the other hand, 5 DoF simplifications cause the missile to be initialized with the required large sideslip angle and small angle of attack, because the roll degree of freedom has been suppressed.

This different kinematic behavior is also evident in the yaw seeker gimbal angles. The highly banked missile exhibits very little initial yaw gimbal angle, while the simplified 5 DoF representation starts out with a correspondingly large value.

Naturally, the pilot is not aware of these missile parameters. He is more interested in the intercept time. The endpoints of **Figure 6** show that the missile time-of-flight of the two versions is very close, within about 1 percent.

For a broader look, a typical LAR-3 (see **Figure 2**) envelope is displayed in **Figure 7**. Throughout the envelope the flight times agree well, with the trend that the 6 DoF exhibits slightly longer times. This tendency is attributed to the fact that the initial transients, as more accurately duplicated by the 6 DoF, delay the trajectories slightly. The 2.65 sec contour was also included, because it marks the burn-out of the rocket motor.

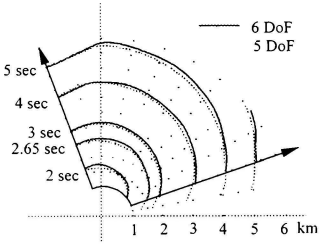


Figure 7 LAR-3 Time-of-flight comparison

Miss Distance Envelope Comparison

The other key factor for the pilot is to know whether the missile achieved a hit and if the target aircraft was eliminated as an attacker. For our study, miss distance serves as criterion, although the damage to a particular aircraft

depends also on the missile's warhead and fuze. We assume, somewhat arbitrarily, that a 6 meter miss is the cut-off point for a hit. Any larger values constitute a miss.

The simplifications of the 5 DoF model, affecting the miss distance, are in the aerodynamics, the autopilot and, above all, in the seeker implementation. We replaced the detailed seeker code of the 6 DoF with an ideal seeker, not constrained by the gimbals and the tracker loop. Multiple runs throughout the LAR-3 envelope lead, as expected, to hits everywhere. However, if we use the 6 DoF simulation with all the details, modeling seeker and sensors with their appropriate noise sources, the envelope changes significantly.

To calculate the LAR-3 envelope for missiles with *stochastic noise*, multiple runs are made against every aimpoint. The miss distances are averaged and displayed as representative terminal miss performance. We used 30 such Monte Carlo runs against each aimpoint, using the automated SWEEP methodology of CADAC.

Figure 8 displays the results of an engagement at 5 km altitude with both aircraft executing 7.5 g maneuvers at .75 Mach.

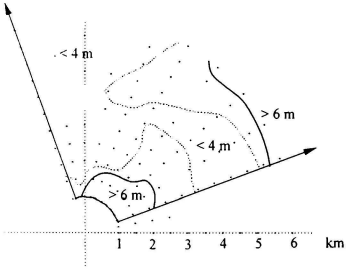


Figure 8 6 DoF LAR-3 envelope; 30 Monte Carlo runs against each aimpoint

Compared to the 5 DoF model, the envelope is reduced for close-in shots – out to 2 km – and at high off-boresight shots beyond 5.5 km. The sweet spot lies at midranges and small look angles. A major reduction in real envelope performance occurs for look angles greater than 70 deg at all launch ranges. It is caused by the realism of seeker gimbal limits. To understand this phenomenon, we need to look at the trajectory more closely.

The maximum look angle of the SR1S seeker is 90 degrees. This allows lock-on before launch throughout the envelope. After launch, there is a delay of 0.25 seconds before guidance is enabled. During this time, the missile tends to align itself with its velocity vector by virtue of its inherent stability. This leads to a horizontal turning away from the target, through an angle perhaps as large as the launch angle of attack, and a corresponding increase in seeker pitch gimbal deflection. Whenever the seeker is locked on before launch at a gimbal angle between 90 and 90-α deg, there is a chance that the weathercock stability of SR1S will call for excessive gimbal deflection, and that a loss of signal will result. This results in the reduced off-axis performance of **Figure 8**. Maximum gimbal deflection is exceeded during the first 0.25 seconds of flight, and the seeker breaks lock on the target whenever the initial look angle exceeds 72 degrees. The kinematic seeker simulation ignores the gimbal stops, and provides whatever deflection is required to maintain target tracking.

The situation depicted in **Figure 8** signifies the physical limitations whenever launches are made at high angle of attack. Such launches can be fairly common in CIC engagements, but are normally very rare at longer ranges.

Many CIC engagements, particularly those which feature high angles of attack at launch, are initiated from subsonic conditions. As a result, the low speed maneuverability of the missile plays a dominant role in determining whether or not a gimbal limit violation is likely to be a problem. The characterization of missile airframes intended for a CIC role should, therefore, include the lowest subsonic speed at which a launch is likely to occur, with a margin to assure that extrapolation of aerodynamic data will not be necessary.

The importance of maximum look angle rests in the fact that, in circle fights, such as LAR-3, allowable look angle may determine the first launch opportunity for the shooter. For the baseline conditions chosen for this study, an 85 deg look angle system could launch nearly a second earlier than a 70 deg system. This could cause a significant change in the overall engagement timeline, and might cause a reversal in system effectiveness evaluations. This disparity between simulations with respect to look angle would be unacceptable, but the

problem is easily corrected . Assuming the 6 DoF simulation to be the truth model, the 5 DoF can be brought into agreement by reducing the limiting gimbal deflection input. A maximum pitch gimbal deflection of 70 deg for the 5 DoF should lead to an envelope with a maximum look angle very close to that of the 6 DoF.

In summary, 5 DoF models, adjusted for 6 DoF effects, can be used in CIC - and of course in BVR - engagements. They portray sufficiently accurate the time-of-flight and the in-range / out-of-range envelopes for the pilot's situation awareness. However, low speed, close-in, and large off-boresight shots require 6 DoF missile models, or, as a minimum, confirmation of such engagements by off-line 6 DoF runs.

Real-Time Conversion

The conversion of batch simulations, used in the design phase, into real-time code has always been a disjointed process. The MIL simulator has its own, tightly integrated, missile fly-out models. In some instances, the same aerodynamic and autopilot subroutines are engaged in driving the aircraft and the missile models, albeit with different data sets. Then, the data tables of the missiles must be restructured to fit the aircraft mold, and sometimes new subroutines must be added to represent missile peculiar features like seekers and sensors.

One can imagine how difficult it is to validate the missile fly-out performance. Many test cases have to be run in both environments and analyzed for inconsistencies. Adjustments will lead to repetitive checks that eventually have to be cut short to prevent schedule slippage.

To avoid these diversions, a *converter program* was developed that starts with the batch simulation, strips it of unnecessary code and translates it into a package that is dropped into the MIL simulation. All missile subroutines remain in tact and only the interface variables need be reconciled. More lines of code penalize this approach, but the execution speed remains the same. This is a small price to pay, considering the abundance of memory in modern computers. The rewards lie in the much simplified validation phase and the assurance that the missiles in the MIL simulator reflect accurately the designer's intent.

This conversion process requires several steps and applies equally to 5 DoF and 6 DoF simulations. After having validated the missile simulation in the batch mode, the programmer runs a test case and records the input to the missile – e.g., target states – on a track file. Then the converter program translates the batch code into a subroutine package, which is validated in the *test bed*, driven by the track file. This code is now ready to be integrated into the *MIL frame* simulator.

The schematic of **Figure 9** depicts this procedure. The missile identification code xx,

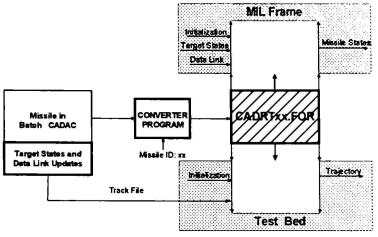


Figure 9 CADAC Converter program

inserted during conversion, attaches itself to every subroutine and labeled common statement. Thus, every missile package becomes unique, and different types of missiles may be flown in the MIL simulator simultaneously.

The communication between the MIL frame and the missile simulation occurs over a common *A array*. It is the exclusive conduit for missile initialization, target state tracking and data link transmission, as well as the missile state feedback. **Figure 10** shows the interface with three missile models x1, x2 and x3.

This integration is currently occurring at IABG, a German MOD research center. Several missile concepts from the U.S., U.K., France and Germany, modeled in CADAC, are converted to code packages for their MIL simulator. The 5 DoF versions are programmed first because of reduced computer loading. However, it is also planned to introduce the 6 DoF models, to study the implications of model fidelity on the pilot's situation awareness.

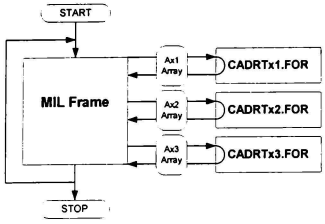


Figure 10 Interfaces between multiple missiles and MIL simulator

Conclusions

We reviewed air combat maneuvers and focused on the close-in combat (CIC) engagement, stylized by the LAR-3 launch acceptable region, as the most demanding situation.

Employing the 5 DoF and 6 DoF simulations of a generic short range air-to-air missile, we assessed the effect of simulation fidelity on the pilot's situation awareness. We concluded that 5 DoF models, adjusted for 6 DoF effects, can be used in CIC and of course in beyond visual ranges (BVR) engagements. They portray sufficiently accurate the time-of-flight and the in-range / out-of-range envelopes for the pilot's situation awareness. However, low speed, close-in, and large off-boresight shots require 6 DoF missile models, or, as a minimum, confirmation of such engagements by 6 DoF batch runs.

The integration of new missile concepts into combat simulators is greatly simplified by a conversion process that leaves the missile code intact, instead of overlaying it on the aircraft model. The results of an ongoing experiment that converts CADAC batch code directly into real time subroutines looks promising.

TOOLS FOR INCLUDING REALISTIC REPRESENTATIONS OF OPERATOR PERFORMANCE IN DOD CONSTRUCTIVE SIMULATIONS

Edward A. Martin, PhD*
Crew System Interface Division
Human Effectiveness Directorate
Air Force Research Laboratory
Wright-Patterson AFB, OH 45433-7511

Bryan E. Brett
Science Application International Company
4031 Colonel Glenn Highway
Beavercreek, OH 45431

David G. Hoagland
Crew System Interface Division
Human Effectiveness Directorate
Air Force Research Laboratory
Wright-Patterson AFB, OH 45433-7511

Abstract

Military weapon systems are normally built to satisfy a set of requirements levied by the warfighter. All these weapon systems are manned in some sense, yet tools for quantifying the effectiveness with which a crewstation must support operator performance are lacking. Analysts and decision-makers need a means to readily model and understand the effects of human performance on total weapon system effectiveness when translating operational requirements into system requirements. This paper discusses the research and demonstration activities being conducted by the Combat Automation Requirements Testbed (CART) Program within the Air Force Research Laboratory / Human Effectiveness Directorate. CART will demonstrate how human-in-the-loop and constructive operator models and data can be integrated with Simulation-Based Acquisition activities for the purpose of defining crewstation requirements. Utilizing the Army's IMPRINT human-performance modeling environment, CART will provide High Level Architecture (HLA) interfaces that enable human-performance models to interact with constructive models of systems. A second extension will incorporate the ability to represent the goal-oriented nature of human performance. Modelers and analysts will be able to define operator goal states and priorities that dynamically drive task network models based on changing states and events in simulated military environments.

---

* Senior Member, AIAA

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

Acronyms

CART	Combat Automation Requirements Testbed
DOD	Department of Defense
DMSO	Defense Modeling and Simulation Organization
FOM	Federation Object Model
HIP	Human Information Processor
HLA	High Level Architecture
IMPRINT	Improved Performance Research Integration Tool
MOE	Measure of Effectiveness
MOP	Measure of Performance
M&S	Modeling and Simulation
ORD	Operational Requirements Document
RCM	Requirements Correlation Matrix
RTI	Run Time Infrastructure
SAM	Surface-to-Air Missile
SBA	Simulation-Based Acquisition
SWEG	Simulated Warfare Environment Generator

Introduction

Deriving system requirements from constructive simulations

Analysts and decision-makers rely heavily on constructive simulations of a system in its intended environment to help translate mission requirements identified by the warfighter into system performance requirements. Within the constructive system simulation, levels of performance on key subsystem attributes are selectively varied and impacts on mission performance are measured. Levels of subsystem attribute performance that yield desired levels of mission performance are identified. These subsystem-attribute performance levels provide the basis for statements of system requirements. A major benefit arising from the use of objective, quantitative requirements is that they provide explicit criteria against which subsystem designs and implementations can be tested. Given that this criterion performance is achieved, there is greater assurance that desired mission performance will be obtained.

The Problem: Current constructive testbeds simulate the operator poorly

While the current state-of-the-art for constructive simulation enables development of most *system* requirements, it does not support development of *crewstation* requirements. Current constructive modeling environments, such as SWEG and BRAWLER, are limited in terms of the range and type of operator activities that can be represented and manipulated. SUPRESSOR, for example, has a 'thinker' component that permits the user to define some decision logic that leads a model to behave differently under different conditions. It does not, however, provide for detailed representation of operator behavior such as sensing information, manipulating controls, or implementing workload mitigation strategies. Another limitation of current models is the extent to which execution under specific conditions can be traced and understood. As an illustration, BRAWLER is a very detailed model in terms of representing what a fighter pilot might do in air-to-air combat. On a given run of BRAWLER, however, it is difficult to trace the execution of model components and understand *why* the components executed as they did. Finally -- hand in hand with the limitations described above -- is the difficulty in obtaining data needed to evaluate performance of such models at a detailed level.

Due to the limitations in effectively modeling the operator, the crewstation has not been considered during the constructive, simulation-based trade studies conducted early in system acquisition. Thus, the crewstation has been omitted from the trade-off process that produces requirements for many other critical subsystems. The result has been that crewstation requirements are developed relatively late in the acquisition process. Crewstation requirements that are eventually developed tend to describe components of the crewstation (e.g., displays of a certain size and resolution and specific types of controls to be used) rather than levels of operator performance that must be supported. In the absence of objective, performance-based requirements levied on the crewstation, it is more likely that crewstations will be produced with flaws that result in substandard mission performance and require remedial action during the production phase.

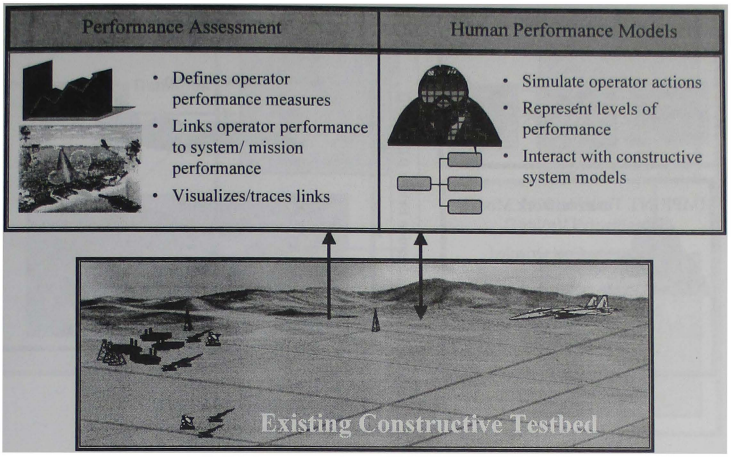


Figure 1. CART Tools for generating crewstation requirements.

System-level performance requirements need to *realistically* consider the effects of operator performance during each step of the acquisition process -- from analysis of alternatives through full-scale production. For example, a military analyst currently associated with a strike-fighter program noted that, "Every single analysis that I have ever seen has suffered from the lack of capturing smart tactics. Mistakes such as pursuing an attack when the tactic should have been 'run away' lead to mission outcomes (aircraft loss) that seem to indicate system deficiencies when in fact the system was misused tactically." Analysts and decision-makers need a means to readily model and understand the effects of human performance on total weapon system effectiveness when translating operational requirements into system requirements, and need to be able to visualize these effects at different levels of aggregation.[1] The technical objectives of the Air Force Research Laboratory's CART Program address these needs.

CART Overview

CART Objectives

The CART program will extend current constructive modeling and simulation (M&S) testbeds by providing new tools for generating crewstation requirements as illustrated in Figure 1. One is a *human performance modeling capability*. With this tool, analysts will be able to create models that simulate activities operators would perform in a system. Analysts also will be able to parameterize the models to reflect different levels of operator capability. These human performance models will be integrated with constructive models of a system and interact with the system in the context of a simulated mission. The second tool provides *performance assessment capabilities*. This tool supports generation of measures of operator performance. Operator measures will be linked to measures of system performance and mission

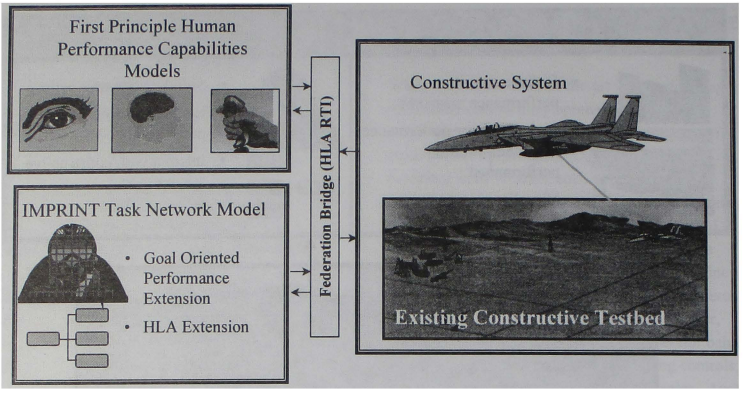


Figure 2. CART's Human Performance Modeling Architecture.

effectiveness. With this tool, relationships between operator performance and system and mission performance can be visualized and traced. Levels of operator performance (MOPs) that are required to produce desired mission outcomes (MOEs) can be identified.

CART human performance modeling architecture

The architecture being used for integrating human performance models into constructive level simulations is shown in Figure 2. The human performance-modeling environment will be a hybrid of two approaches to human performance modeling: *task network modeling* and *first principle modeling*. Task network modeling will be the core human-performance modeling method. Task network modeling breaks the human performances of interest into a series of tasks characterized in terms of performance times, accuracy, and probabilities. Tasks are linked together into networks that represent sequences and paths performance can take. Within CART, the IMPRINT tool mentioned earlier will be used to provide baseline task network modeling capabilities. IMPRINT will be extended to provide representation of the goal-oriented nature of human performance and to communicate with external models via the HLA. While task network models provide an easy-to-understand representation of human performance, their fidelity is limited in terms of modeling specific human capabilities such as cognition and perception. For this reason, users will have the ability to augment the task network models with first-principle models that provide high-fidelity representations of human capabilities. Essentially, tasks in the task network will call first-principle models that represent the capabilities required in the task. The first-principle model will execute the capability and return the parameters required by the task network model.

The DMSO's HLA will provide the communications link between models. Data will be passed between architecture components using the HLA RTI. The task network model will receive data regarding system and mission status from the constructive system simulation and data about the external world (e.g., SAM launches) from the mission environment models. Actions to be implemented by the system (e.g., maneuver, target designation, weapon launch) will be passed to the constructive simulation by the task network model.

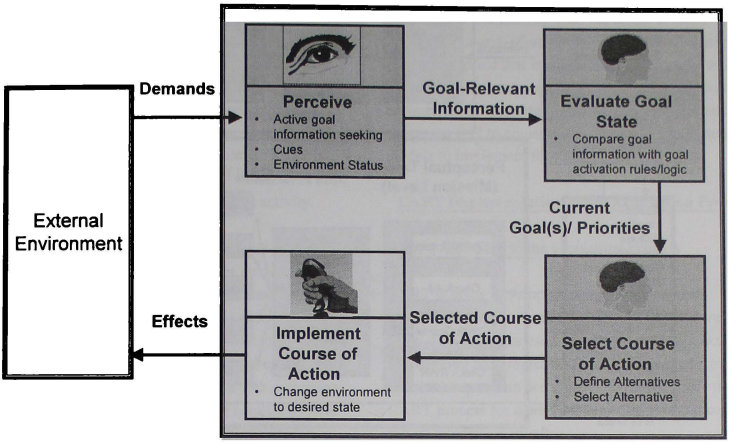


Figure 3. Basic Human Information Processor Model (after Hendy and Farrell, 1997)[2]

Similarly, the task network model will use the RTI to pass data to first principle models to initiate them and receive the results of first-principle model execution.

Modeling Theory and Methods

Underlying assumptions and theory

CART's approach to human performance modeling is based on two fundamental assumptions. The first is that operator success is achieved by meeting mission performance demands that are levied by factors external to the operator. This is consistent with our view of the operator as a constrained component of the system. The demands are, in effect, constraints. If the operator does not perform within the constraints (meet demands) mission performance can be degraded. The second assumption is that, in meeting demands, the operator functions as an information-processor.[2] That is, the operator identifies current demands and selects and implements courses of action to meet those demands. In this process the operator seeks information from the environment about demands and applies information held internally and additional information from the environment to meet those demands as illustrated in Figure 3. Inherent in the information processor are capabilities and limitations that interact with demands and that can affect mission outcomes.

In order to successfully meet mission demands, the operator must determine which demands are impinging at a point in time, prioritize them if there are multiple demands, and then act to meet those demands. Demands from the environment are first perceived by the operator using one of the five senses. Perception of demands is an active process in which the operator purposefully seeks specific information about current demands. This information seeking is focused and methodical, based on training and experience. Given that multiple demands can be active at the same time, a mechanism is needed to sort among concurrent demands to chose which one(s) get serviced first. The model assumes that in a given system-mission environment the operator has an internal goal structure that helps him assess and prioritize demands to be met. These goals are associated with functions that must be

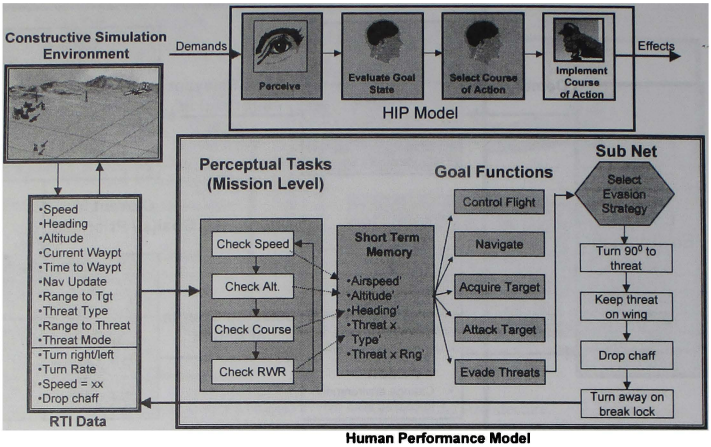


Figure 4. Representing the Human Information Processor (HIP) with IMPRINT.

performed successfully to accomplish the mission. The goals are defined in terms of states of the external environment that the operator seeks to control. In goal-state evaluation, information from the environment provided by perceptual processes is compared with internally held knowledge of expectations about world states and rules for determining when a goal state becomes active. When goals become active, attention turns to selecting a course of action for bringing the current state of the world into the desired state. Course-of-action selection can involve a variety of levels of cognitive processing (e.g., skill-based, rule-based, or knowledge-based reasoning). It can also involve other perception and action components that are applied to gain additional information needed to select a course of action. Once a course of action is selected, it is implemented and its effect on the environment is observed. Course-of-action implementation generally involves motor activity (e.g., manipulate a control or throw a switch). Observation-of-effect is performed by the perceptual capabilities, which, in turn, feed the goal states and the cycle repeats itself until the desired state is achieved.

Readers familiar with control theory will recognize that the information processor model is really a type of a closed-loop control model.

Representing the human information processor (HIP) model with IMPRINT

The model implementation is shown schematically in the lower half of Figure 4, and corresponds to the HIP model that is shown above it in the figure.

The constructive simulation environment (i.e., the models of the system and mission environment entities) will provide the representation of demands to the information processor. Passage of data between the human performance model and the constructive testbed will be controlled by the HLA runtime infrastructure (RTI). IMPRINT collects 'demands' data from the RTI and stores it in user-defined variables.

Information seeking about current demands in the mission environment will be performed by a network of tasks representing perceptual tasks. The network will control the sequence and timing according to which the operator 'observes' displays and instruments and 'listens'

for communications, tones, alarms, etc. Perceptual information seeking is driven by information needs for evaluating goals. Perceptual tasks will update other user-defined variables that represent short-term memory. This reflects the fact that perception is a constrained process -- we cannot know everything about our world instantaneously. What we know is determined by when it is perceived -- which is, in turn, controlled by our 'schedule' of perceptual activity.

Goal states are evaluated from the contents of short-term memory. Thus, conditions in the environment can change such that a goal should become active, but the goal will not actually trigger until that new condition is perceived and reflected in short term memory. Goals evaluate on every cycle of the model. Initiating-condition expressions provided by the user determine when a goal triggers. Once initiating conditions have been met, additional logic provided by the user in regard to goal priority and activation relative to other higher-priority goals determines whether the goal actually becomes active. The goal state 'evade threats', for example, would be triggered when threats were present and within a certain proximity to the aircraft. Because threat evasion is such a high priority goal, the model developer might decide to suspend activity under any other goal state that might be triggered while the evasion goal state is active.

Task sub-networks are activated for goals that become active. Within these sub-networks, decision nodes can be provided that represent alternative courses of action. Within the decision node, logic can be specified that selects the course of action best suited for the current circumstances. When decision-making is complex, course-of-action selection itself might be represented by a network of tasks. Under the threat-evasion example, selecting a course of action would probably involve considering the type of threat and choosing from among a set of evasion options (e.g., maneuver, countermeasures, or a mix of both).

As the course of action executes, inputs to the constructive system are provided via updates to user defined variables that, in turn, update variables in the RTI. The constructive system model receives this data from the RTI and then changes its performance accordingly. Continuing with the threat evasion example, a course-of-action implementation strategy might be applied that involves maneuvering the aircraft while applying some countermeasures. As the task network model executes these actions, data are sent across the RTI to command the constructive system models to implement the corresponding actions.

CART Implementation in the Acquisition Process

What CART adds to the acquisition process

CART is being developed to support the system acquisition process by enhancing the constructive simulation activities used to explore system concepts and develop requirements. Once the needs of a given acquisition program are thoroughly understood, the CART process for achieving human-performance-model integration involves:

- decomposing a system's mission to understand the various human and system tasks,
- developing human performance models that characterize human behavior on the tasks and that will interface with the existing constructive environments,
- collecting and analyzing the data to identify levels of task performance that determine mission success,
- translating the findings into crewstation requirements that state the levels-of-performance that the crewstation must enable.

The end result of CART implementation in the acquisition process is the development of *crewstation performance requirements* that will supplement the higher-level system capability requirements.

Role of the Requirements Correlation Matrix (RCM)

Throughout the acquisition process, a formal record of evolving system requirements is contained in the Operational Requirements Document (ORD). Within the Air Force, these requirements are also summarized in the RCM. Developed during concept exploration and refined during subsequent phases of the acquisition process, the ORD and it's accompanying RCM

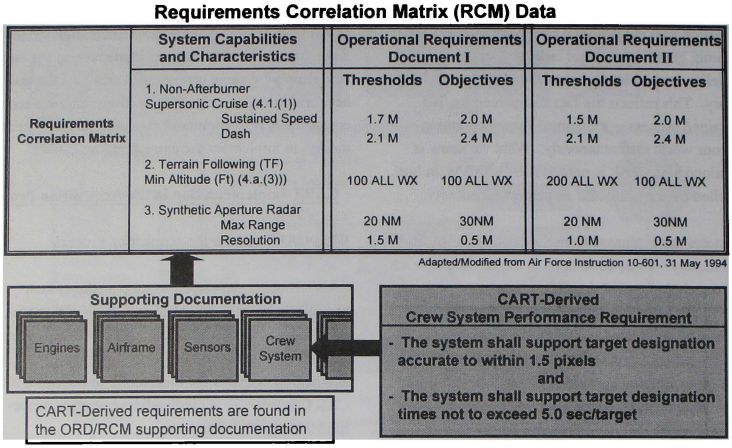


Figure 5. CART's role in supporting the requirements-specification process.

formally state the performance and related operational parameters for the proposed system. Shown in Figure 5 are components of an RCM for a hypothetical strike fighter aircraft. Requirements are stated in terms of *operational* capabilities and characteristics that the aircraft must exhibit. Notice that for each stated capability or characteristic, there are both "threshold" and "objective" values listed. A *threshold value* reflects the minimum acceptable operational performance for the proposed system. The *objective value*, however, represents a higher degree of capability that would lead to an identifiable increase in operational performance. In this example the threshold value for sustained, non-afterburner flight is Mach 1.7, whereas the objective value is Mach 2.0. Since the ORD and RCM are living documents that evolve over time, they also include the ongoing revisions to the requirements. Notice that ORD II data represent a change in requirements from ORD I. The threshold value for sustained non-afterburner flight has been revised downward to Mach 1.5. Supporting rationale for both the initial requirement and subsequent changes to the requirement are documented. These justifications and rationale are often based on the results of a specific tradeoff analysis or trade study examining the relative impacts of various levels of a given parameter on mission outcomes.

Documentation of CART-derived requirements

The goal of the crewstation performance requirements generated using CART is to provide a performance benchmark against which crewstation designs can be evaluated. For example, a simulation-based trade study examining alternative air-to-ground radar systems for the strike fighter might conclude that a key determinant of mission success is the operator's ability to quickly and accurately designate a target aimpoint on the radar image. Examining target-designation task performance and tracing this performance to mission outcomes, CART data analysts could identify specific levels of designation-performance that the crewstation must support in order to achieve the desired mission outcomes. In the example of Figure 5, it is determined that cursor designation must be achieved within 1.5 pixels of the desired aimpoint and take 5 seconds or less.

It is not anticipated that CART-generated requirements such as these will be stated explicitly at the level of the

ORD and RCM. Rather, they will be passed to the system designers with the lower-level trade study documentation that supports the ORD and RCM. By providing these performance-based crewstation requirements relatively early in the acquisition process, the CART approach promises to result in the faster, less-costly acquisition of more effective manned systems.

Discussion and Conclusion

As M&S technologies continue to make rapid advancements, their utility and value to the acquisition process continue to increase. However, there is still room for improvement in both the tools and processes being implemented to support acquisition. It is generally accepted that -- in order to reap the full benefits of M&S -- we need to develop increasingly valid simulation environments and to better share these environments and data within and among acquisition communities.

Representing the current framework for M&S in acquisition, the Simulation Based Acquisition (SBA) vision identifies three goals for enhancing the acquisition process:

- reduce the time, cost, and risk associated with acquisition,
- increase the quality of the resulting systems,
- enable integrated product and process development.[4]

The DOD M&S Master Plan identifies *six specific objectives* that will help achieve these goals:

1. develop a common technical framework for M&S,
2. provide timely and authoritative representations of the natural environment,
3. provide authoritative representations of systems,
4. provide authoritative representations of human behavior,
5. establish an M&S infrastructure to meet developer and end user needs,
6. share the benefits of M&S.[5]

CART will help realize Objective 4 of the DOD M&S Master Plan by providing the capability to readily integrate models of human performance into the constructive modeling activities that address overall system performance. CART-developed methods and tools will be used to integrate human-performance representations into early constructive simulation activities and to help generate human / system performance requirements that a proposed system must support. These requirements will then help focus crewstation design activities, enable early identification of potential crewstation design problems, and provide performance-based standards against which crewstation designs can be tested. By incorporating a representation of the human operator and demonstrating the operator's impact on mission performance, a better representation of the total system will be provided and judicious acquisition decisions regarding total system requirements can better be made. In so doing, CART implementation will support the SBA objectives of reducing acquisition time, cost, and risk while increasing system quality and effectiveness.

References

[1] D.G. Hoagland, E.A. Martin, & B.E. Brett, *Improving the DoD's Requirements Process Through Inclusion of Realistic Operator Performance*, Simulation Interoperability Workshop Paper Number 98F-SIW-071, Fall 1998.

[2] K.C. Hendy & P.S.E. Farrell, *Implementing a Model of Human Information Processing in a Task Network Simulation Environment*, DCIEM Number 97-R-71, Defence and Civil Institute of Environmental Medicine, North York, Ontario, Canada, December 1997.

[3] J. Rasmussen, A.M. Pejtersen, L.P. Goodstein, *Cognitive Systems Engineering*, John Wiley & Sons, New York, 1994.

[4] Office of the Undersecretary of Defense for Acquisition & Technology, *Simulation, Test, and Evaluation Process (STEP) Guidelines*, December 1997.

[5] Defense Modeling & Simulation Office, *Modeling & Simulation (M&S) Master Plan*, October 1995.

ARE CRITERIA FOR MOTION CUEING AND TIME DELAYS POSSIBLE?

Ruud Hosman
Director, Aerospace Man-Machine Systems Consulting
Wielengahof 46, 2625 LK Delft, The Netherlands
r.hosman@amsconsult.demon.nl

Abstract

In spite off a continuing development of flight simulation over the past thirty years, there are still some areas where flight simulation is not a one to one replacement of real flight and can not completely fulfill the reasonable and objective requirements. To fulfill these needs, the proper feedback of effective motion cues seem to encounter two significant shortcomings, resulting from a lack of full understanding of (a) the impact of motion cueing on the pilot's behavior and (b) the requirements for motion cueing in the specific training application.

During the past twenty years, research projects on pilot motion perception and manual control have been performed in the Netherlands at the Delft University of Technology, the National Aerospace Laboratory, NLR, and the TNO Institute for Human Factors. The results have not only improved the knowledge on pilot's aircraft motion perception and control, but also initiated a reconsideration of motion feedback in flight simulation. Full flight simulation is meant to integrate the pilot's skill-based, rule-based and knowledge-based behavior in his control of the total aircraft system. Distinguishing the contribution of motion feedback to these three levels of behavior provides the tool to discriminate the impact of motion feedback on these levels of the resulting pilot behavior. Based on this discrimination, a review of motion system requirements, and washout filter design and optimization, subject to the training goal, becomes possible. The paper reviews the major results of the motion perception research, explains the discrimination of motion cues based on the three levels of behavior, and shows the impact on motion-base drive algorithm design. The significance of simulation-induced delays on compensatory manual control is shown, underscoring the value of such research in objectively defining future simulator requirements.

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

1. Introduction

Since the introduction of six-degrees-of-freedom motion systems in the 1970's, flight simulation has developed to a technology of nearly full maturity. The evolution of flight simulation has been primarily based on a technological developments: Faster computers, better and more extended software, smooth, high-resolution images projected over wide fields-of-view, and advanced motion systems have all improved the quality of flight simulation significantly. Based on these developments, simulator operators and the regulatory authorities have accepted the flight simulator as the ubiquitous training tool for transport aircraft pilots. This acceptance is in spite of the fact that the Full Flight Simulator (FFS) is not accepted as a *one-to-one replacement* of the real aircraft. In the near future, however, it may be expected that in-flight training on transport aircraft will become unacceptable for economical, environmental, and airport capacity reasons. On the other hand, training of flight crews will become increasingly dependent on flight simulation since more and more flight procedures and abnormal conditions will somehow have to be trained. The reliance on simulation can be demonstrated through the recent and growing introduction of zero flight-time (ZFT) type-conversion training for young and less-experienced pilots entering the major airlines in Europe. Because the operators are in need of simulation-based training that can fully replace in-flight training for these of activities (and where the FFS is not yet accepted as a replacement of the real aircraft), flight simulation needs a further improvement.

On the other hand, the FFS seems to be, in some applications, an overkill, or may be simply too expensive. In those cases, a Flight Training Device (FTD) may be a cost-effective alternative to the FFS. Finally, there is a strong need to improve the safety of daily airline operation. An improvement of pilot proficiency is one of the pillars towards achieving that goal.

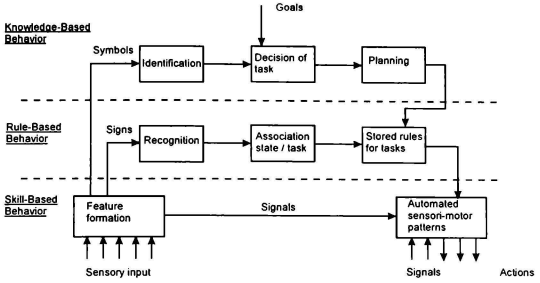


Figure 1. Simplified scheme of the three levels of behavior of the human operator (Rasmussen, 1983).

There are two important differences between the aircraft and the simulator: The aircraft, when compared to the simulator, has an almost unlimited freedom of displacement in six degrees-of-freedom. In other words, there is a big difference between the dimensions of the aircraft and the simulator motion spaces. During daily operation of the aircraft and the modern simulator, the most important differences in motion freedom are the three linear displacements and the angular displacement around the yaw axis. The angular displacements in pitch and roll around the longitudinal and the lateral axes are not equal, but the same relative magnitude can be achieved. In the simulation process the aircraft motions are transformed to the simulator motion space by the motion drive algorithms, or washout filters (both terms will be used synonymously in this paper). Their aim is to provide approximated aircraft motions to the simulator cab, while keeping these motions within the kinematic limitations of the motion-base mechanism.

A second difference between the real aircraft and the simulator is the fact that the dynamics of the latter are represented through equations of motion, solved by a digital computer. The computed state of the simulated aircraft is the reference by which the presentations with the simulated flight instruments, the outside visual system, and the simulator motion system (through the motion-drive algorithms) are generated. Since all these systems are based on digital computing, and since these digital processes require finite computation time, these in turn induce simulation time delays.

In this paper attention will be given to the impact on the interaction between the pilot and the simulated aircraft of (a) the transformation of the aircraft motions to simulator motions, and (b) due to the simulation time delays. As a result of these simulation-induced properties, the changes to the pilot's perception of the

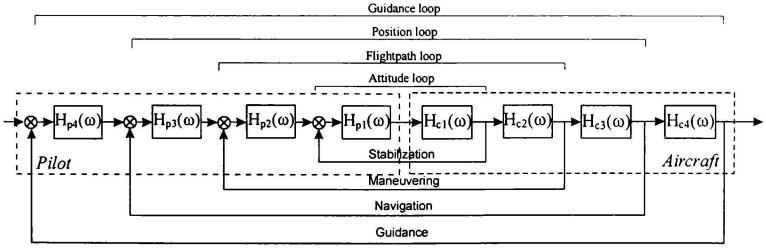


Figure 2. Nested control loops for pilot's aircraft control.

aircraft motions, and to his control behavior due to simulation will be considered. At several research institutes in the Netherlands (the National Aerospace Laboratory, the Delft University of Technology and the TNO Institute for Human Factors), a number of research projects on pilot control behavior and self-motion perception have been performed. The results of this research give a broad insight into the influence of visual and vestibular stimulation on motion perception and control. Based on these results, a better understanding of the role of visual and vestibular motion feedback on pilot control behavior has been developed. In addition, the influence of simulation can be better understood, and can also help to develop criteria for motion cues and time delays.

2. The pilot's task and its reinforcement

The pilot's task in a modern transport aircraft represents much more that of a management (hence supervisory control) than of manually controlling a dynamic system. As a supervisory controller, the pilot directs and/or controls the different aircraft systems and is free to decide which part of the task should be performed manually or by the automatic systems. Rasmussen (1983) described the human operator behavior through three levels: skill-based, rule-based and knowledge-based behavior (see Figure 1).

At the skill-based level, the perceptual-motor system acts as a continuous control system. The outputs of the sensors, in this case the visual system, the vestibular system, proprioceptive system, etc., are perceived as continuous ***signals*** as a function of time and used to generate the control output.

At the rule-based level, the perceived stimuli are considered as ***signs***, marking changes in conditions and/or procedure. Based on the signs, the new rule is selected. After selection of the rule, the execution is performed with the learned automated sensory-motor patterns or skill-based behavior.

Functional reasoning to predict future changes or behavior of the environment is based on ***symbols*** at the knowledge-based level.

Thus, knowledge-based behavior is based on a cognitive understanding of the particular task and results in conscious logical decisions; rule-based behavior is based on rules and procedures; skill-based behavior is generated by trained sensory motor patterns. When this classification is applied to the pilot's task, then a better understanding of how the pilot performs his control task can be obtained, as well as how different levels of simulation can be used in pilot training, and what requirements should be set.

Knowledge-based behavior is based on information, that can be learned for a great deal in the classroom and/or by experience. Rule-based behavior can be trained with part-task trainers such as procedure trainers. Manual control skills, on the other hand, have to be trained with the correct aircraft characteristics. Therefore, different levels of behavior have to be trained in different ways and with different tools. The pilot's total task, however, has to be trained in a realistic full flight environment to make a *full integration* of the different levels of behavior possible. This is the main reason why pilot training is currently completed in full flight simulators or, when such facilities are not available, in real flight.

As far as aircraft control from departure to destination is concerned, the pilot performs the control task in a closed loop. Due to the higher-order nature of aircraft dynamics, the pilot is not able to control the aircraft position directly. Instead, the pilot performs the control task in a number of steps or "nested control loops" (see Fig. 2). In the case of conventional aircraft control, the inner loop controls the aircraft attitude, while the most outer loop is involved with the guidance of the aircraft from the departure to the destination airport. In the inner attitude control loop, the pilot performs a skill-based attitude-tracking task, while he performs a knowledge-based task in the outer guidance loop. Going from the inner loop to the outer loop, a pilot's behavior changes gradually from skill-based, via rule-based, and finally to knowledge based behavior. Besides the pilot's control task, there are quite a number of additional tasks that require primarily rule-based and knowledge-based behavior.

For effective training, it is also therefore important that the pilot in the simulator behaves at all three levels of behavior for the full-simulated flight envelope in the same way as in the real aircraft. However, given the fundamental differences between the aircraft and the simulator (as mentioned earlier), it is important to establish how the deficiencies of the simulator interact at the three levels of behavior.

Prior to presenting information on that subject, research results on visual-vestibular interaction in perception and control will be summarized.

3. Research results

During the past twenty years, research projects on pilot perception and control of aircraft motions have been performed in order to gain a better understanding of the influence of the visual-vestibular stimulation on motion perception in general and, in particular, to pilot's perception and control behavior. The results of three research projects will be reviewed: the influence of visual-vestibular stimulation on skill-based tracking behavior (Hosman, 1996, 1998), and the perception of self-motion (van der Steen, 1998, Mesland, 1998).

Skill-based control behavior

In the late sixties and the seventies, a number of studies were performed to prove the positive influence of motion feedback on pilot tracking performance and control behavior (Meiry, 1965, Newell, 1968, Stapleford, 1969, Junker and Replogle, 1975, Levison and Junker, 1977). The results of these studies showed that motion feedback has a positive influence on a pilot's tracking performance, and furthermore supported the application of six-degrees-of-freedom motion systems in flight simulation. These studies, however, did not explain why and how motion loops (Hess, 1987). Because motion feedback has an important influence on the performance and the bandwidth of the inner loop, it also has an indirect influence on the bandwidth and performance of the outer loops.

By minimizing a cost function, the model can be adapted to the particular aircraft dynamics. Then, by applying the model to the control of the simulated aircraft, the influence of the motion feedback, the motion drive algorithms, and the time delays can be studied, by adapting the model to the changes in the closed control loop with the simulated aircraft.

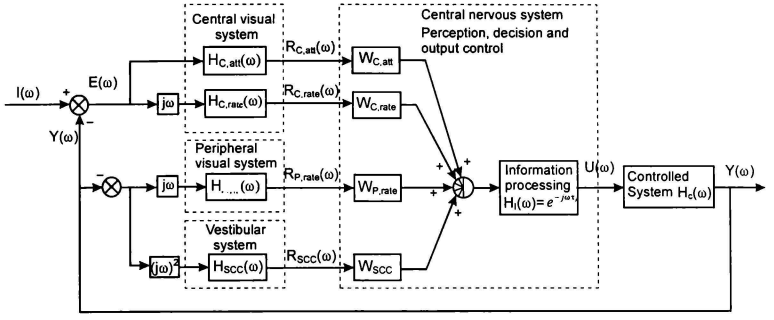


Figure 3. The descriptive pilot model in a tracking task (maneuver motion) control loop.

feedback improved pilot's control behavior.

Hosman (1988, 1996, 1997) showed that the faster response of the vestibular system compared to the visual system response to angular rate input is the main cause of a pilot's improved control behavior and performance in attitude tracking tasks in the presence of motion feedback. Hosman (1996) postulated a pilot model incorporating the individual visual and vestibular sensors and describing skill-based control behavior in tracking tasks. The parameters, W_i, weighing the sensory output in generating the control signal, are the only free model parameters, Fig. 3.

In the inner attitude control loop, Fig. 2, the pilot's control task can be considered a tracking task (maneuver motion) in which the pilot must follow the input attitude reference and, secondly, a stabilizing task (disturbance motion). In the latter case, the pilot compensates against external disturbances. The bandwidth of the inner loop is the highest and, as a result, determines the maximum bandwidth of the outer

Self-motion perception

Besides the influence on the pilot's tracking behavior, visual-vestibular stimulation affects a pilot's perception of self-motion, which is equal to the movement of the aircraft in its environment. Note that the perception of the aircraft motion by the pilot is required in order to perform his rule-based and knowledge-based behavior in the middle and outer loops of his control task (see again Fig. 2). In these loops, the visually presented information becomes progressively more important when going outwards from the inner attitude-loop to the outer guidance-loop. Correspondingly, the pilot behavior changes to a more *passive* perception of self-motion.

A cooperative research program, which was supported by the National Aerospace Laboratory NLR, investigated self-motion perception and was performed by the Delft University of Technology and the TNO Institute for Human Factors,.

In these experiments, visual and vestibular stimuli with different amplitudes were presented to the subjects.

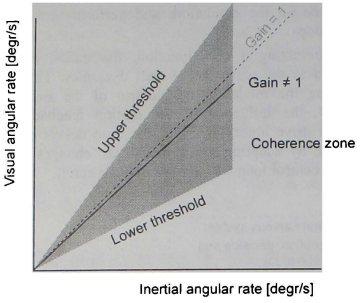


Figure 4. The coherence zone as a function of the visual and vestibular stimulation.

Several techniques were applied to determine if the subjects perceived the visual and vestibular cues coherently (i.e. that they perceived them to be moving at equal rates). A wide variety of linear and rotational motion stimuli (step, ramp and sinusoidal) were presented to subjects to determine the thresholds at which the subjects would detect that there was a difference between the visual and the vestibular motion. To determine these thresholds, the amplitude of either the vestibular stimulation or the visual stimulation was varied. In this way, an upper and a lower threshold was found. The zone *between* the thresholds is called the "coherence zone" (van der Steen, 1998), Fig. 4.

The coherence zone can be determined for linear as well as for rotational velocity. Fig. 4 shows that, for a certain visual stimulus amplitude, the amplitude of the vestibular stimulation may vary between the bounds of the coherence zone while the subject does not notice the difference, and visa versa. Until now, experiments have been performed with linear motion in surge, sway and heave, and for rotational motion in roll and yaw.

The experimental apparatus used for the experiments varied between the research installations: a linear sled was used at TNO, and an optokinetic drum was employed at the Grosshadern Klinikum of the Munich University Hospital. Two research flight simulators were also used: the Delft Universty of Technology facility with a three-degrees-of-freedom motion system equipped with peripheral monitor displays, and the six-degrees-of-freedom motion-based system with a wide-angle dome visual system at NLR).

4. The influence of flight simulation on pilot control behavior.

Given the present state of the art, there are still a number of questions concerning, for example, the quality of simulation (Ray, 1996), the need of motion feedback (Kingston et al. 1996?), and the requirements for effective ab-initio zero flight-time training (Teunisssen, 1997). All of these questions, and the subsequent hiatus of answers, are directly related to the yet limited understanding of the perception and control behavior of the pilot, the generation of the visual and vestibular cues in flight simulation, and the lack of criteria for the simulation of motion cues.

Given the understanding in a pilot's perception and control behavior gained from the research projects described above, an effort was undertaken to determine

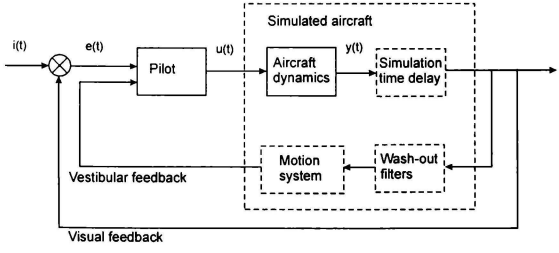


Figure 5. The pilot-aircraft closed loop with a lumped simulation time delay, motion system, and washout filters added to the control loop in the case of simulation.

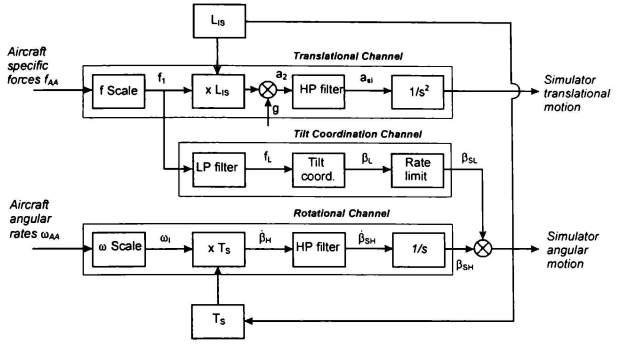


Figure 6. Structure of the high and low pass filters in the classical washout filter.

how these results could be applied to establish which cues are required to support pilot behavior at a specific level or task. As a result of this work, an initial yet significant insight into the establishment of the criteria for motion cues has been realized.

Skill-based attitude control behavior

As shown earlier (Fig. 3), skill-based control behavior is of particular importance in the manual aircraft attitude control, and motion feedback has primarily a direct influence on a pilot's tracking behavior and tracking performance for both disturbance motion and maneuver motion in the simulator.

In Figure 5, a block diagram is presented where in the case of the simulated aircraft a "lumped" effect of simulation time delay and motion feedback via washout filters and the motion system is incorporated into the pilot-aircraft control loop. The influence of a time delay in a control loop, a phenomenon that is well known and well documented in literature, is to decrease the phase margin. Also, if the system stability is degraded significantly, the crossover frequency has to be decreased by lowering the gain by the controller (the pilot). To optimize then the behavior of the pilot-aircraft control loop in the simulation, the pilot will have to adapt his control behavior. In effect then, the pilot will perceive an interaction with a system that is

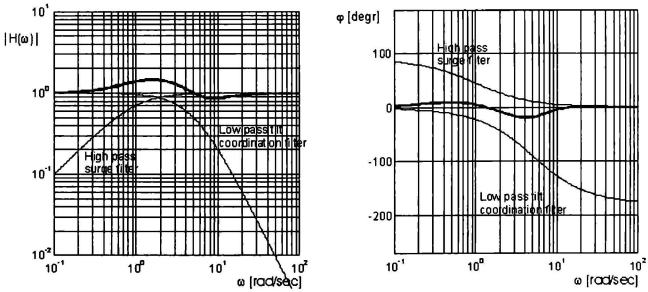


Figure 7. Bode plot of the transfer function of the classical washout filter for the pitch attitude (Nahon and Reid, 1990).

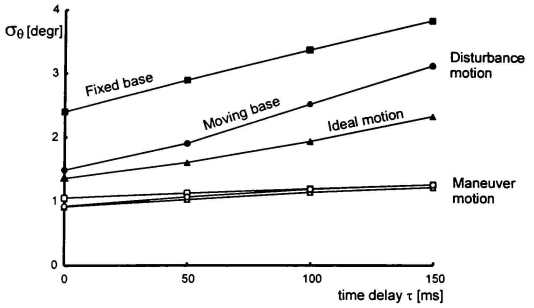


Figure 8. The influence of motion feedback, washout filters and time delay on pilot's tracking performance for disturbance motion and maneuver motion based on the descriptive model.

different than the real aircraft.

To generate realistic simulator motion cues, the washout algorithm (see Figure 6) simulates angular motion in pitch and roll by controlling the tilt coordination, and the rotational channels (Nahon and Reid, 1990). The impact of the washout filters on the control loop is, however, not so clear. A bode plot of the washout transfer function for pitch attitude, shown in Fig. 7, indicates that in the crossover region (1 to 4 rad/sec) the gain is increased, while the phase is decreased. This decrease in phase has a negative effect on the stability of the attitude control loop.

The pilot, faced with these changes of the attitude control characteristics of the simulated aircraft (due to both the effects of the washout and time delay), will adapt his control behavior to compensate for these changes and maintain the best possible characteristics of the pilot-simulated aircraft control loop.

To provide an impression of what the adaptation of a pilot's behavior to the simulated aircraft characteristics means, the descriptive pilot model, Fig. 3, has been adapted to a number of closed-loop control configurations. The aircraft dynamics used were representative of a small business jet aircraft (the Cessna Citation I). The classical washout filters (Nahon and Reid, 1990) were applied to this model, and the time delay was varied between 0 and 150 milliseconds.

The entire model was then adapted to the control loop with a least square optimization procedure with a fixed cost function. The results are presented in Fig. 8.

Figure 8 shows that, in the case of maneuver motion, the pilot performance is negligibly influenced by the motion washout feedback and time delay. For disturbance motion, however, the combined effect of washout filtering and time delay degrades performance considerably. In the fixed-based configuration (hence, no motion), the performance is degraded significantly, just as one might expect.

For both the maneuver motion and the disturbance motion, the pilot, as represented by this model, indeed adapts to the changes in the control loop due to the motion feedback, washout filter, and time delay. Without this adaptation, the closed-loop performance would have degraded much more, and control would have become unstable. Besides the changes in the overall gain of the model, the weightings of the angular rate perceived through the visual and the vestibular systems are also adapted (see Table 1).

Table 1. Pilot model parameters W_{Catt}, W_{Crate}, and W_{SCC}, disturbance motion tracking performance σ_θ, closed-loop crossover frequency ω_c and phase margin φ_m (fixed base and moving base with 100 ms time delay).

	W_{Catt}	W_{Crate}	W_{SCC}	σ_θ (deg)	ω_c (rad/s)	φ_m (deg)
Fixed base	0.54	0.07	0	3.36	3.59	25
Moving base	0.74	0	0.20	2.51	3.84	19
Aircraft	1.42	0	0.26	1.36	4.90	35

Besides the changes in tracking performance, the model weighting factors for the visual-perceived pitch attitude W_{Catt} and rate, W_{Crate}, and vestibular-perceived rate W_{SCC} change considerably due to the three sets of conditions: Fixed base, simulation with time delay and the real aircraft. Due the differences between the aircraft and the simulator, W_{Catt} approximately doubles, and the relation W_{SCC}/W_{Catt} changes from 0.18 to 0.27. (Note that the results of Table one are based on f and ω scales equal to 1, Fig. 6.) These changes demonstrate that a well-trained human being is able to adapt his behavior to new or altered manual control systems. For example, in every day life, we are able to quickly "adapt our control behavior" to the control characteristics of a rented car, a stolen bike, or other vehicles and systems with which we are initially unfamiliar.

The experienced pilot with a well developed skill-based control behavior and familiar with the aircraft characteristics is quite able to *adapt* his control behavior to the particular properties of the *simulated* aircraft, such as those due to motion filtering and time delays. This adaptive nature was confirmed by the Delta-Embry Riddle study (Kingston et al., 1996). The *inexperienced* pilot on the contrary, who gets his first type conversion training to a transport aircraft, is required to somehow learn the necessary skill-based behavior for the control of that large aircraft. The characteristics of a larger aircraft will likely differ considerably when compared to the training aircraft he has flown so far. With current simulation technology, given the need to adapt to achieve adequate control of the simulated aircraft, this trainee may then develop a skill-based behavior which, although is quite sufficient for the simulated aircraft, is not the same for the real aircraft, (Teunissen, 1997).

Rule-based and knowledge-based behavior.

As far as rule-based and knowledge-based behavior is involved in the pilot's task, it is important that the pilot has a correct perception of the aircraft motion state within its environment. Pitch and roll rotations are of direct importance for skill-based attitude control as was shown earlier. In addition, all the six degrees-of-freedom play a role in the outer control loops. Therefore, one can conclude that the perception of the motions in all six degrees-of-freedom is of direct importance for the rule-based and knowledge based behavior.

Since the linear motion space of the simulator motion system is several orders smaller compared to that of the aircraft, there exists an insurmountable difference between the simulator and the vehicle. This is partly compensated by the tilt coordination channel of the washout filters, a process by which the sensation of a longitudinal acceleration can be obtained. As an example, the real and simulated specific forces, f_x, along the longitudinal X-axis during the first seconds of a take-off run are presented in Fig. 9. The first peak in the simulated f_x results from the response of the high-pass filter in the translation channel to the step change of f_x after releasing the brakes. Thereafter, the simulated specific force results from the gravity component due to the tilt of the simulator cabin. The low-pass filter of the tilt coordination channel generates this tilt. The cabin pitch rate is limited to 3°/sec to prevent the pilot from perceiving the tilt rate of the platform.

Fig. 9 shows a large discrepancy between the specific force f_x in the aircraft and the simulator. However, in performing the simulated take-off, the pilot will not perceive this discrepancy. This is primarily a result of the fact that two factors influence a pilot's perception

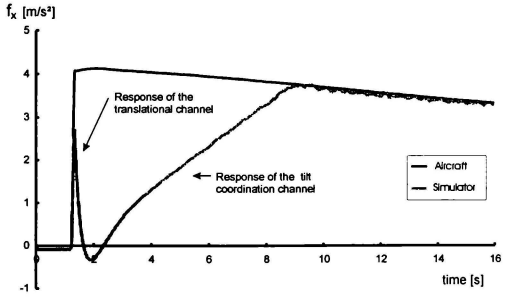


Figure 9. Aircraft and simulator specific force f_x along the longitudinal axis during the first seconds of a take-off run.

Besides, the geometries of motion-base mechanisms can themselves be optimized to be able to present the required motion cues and, in this way, new derivatives of the current motion-base mechanism can be developed (Advani 1998).

6. Conclusions and recommendations

In summary, the present understanding of the visual-vestibular interaction on pilot perception and control behavior, and the distinction between the different levels of behavior, provides the knowledge for further goal-oriented developments in flight simulation. These developments should pave the way towards a more differentiated use of (fixed-base) Flight Training Devices, and Full Flight Simulators, as well as to refinements of classical washout filters and, moreover, providing an impetus to decrease simulator time delays.

The differentiation between the use of Flight Training Devices and Full Flight Simulators must be based on objective criteria, dependent also on the task and associated level of behavior that needs to be trained. However, in the foreseeable future, flight training will always be completed by an integration of the different levels of behavior in a FFS or in the real aircraft. The fine-tuning of washout filters, coupled with a decrease in simulation-induced time delays is required for a noticeable improvement of the transfer of training of skill-based control behavior. The tuning of the washout filters depends on aircraft characteristics, the particular maneuver and/or the region of the flight envelope.

Considering that the results of this research have indicated a need for more definitive studies, and that there is a growing interest and need to refine motion cueing, the time is ripe for further research, development and evaluation of flight simulators from a motion cueing perspective.

References

Advani, S.K. (1998). The Kinematic Design of Flight Simulator Motion-Bases. Ph.D. thesis, Delft University of Technology. Delft University Press, 1998. ISBN 90-407-1672-2.

Hess, R.A. (1987). Feedback Control Models. In: G. Salvandy, Ed. Handbook of Human Factors. John Wiley & Sons, 1987.

Hosman, R.J.A.W.and J.C. van der Vaart (1988). Visual-Vestibular Interaction in Pilot's Perception of Aircraft or Simulator Motions. Proceedings of the AIAA Flight Simulation Technologies Conference. September 7-9, 1988, Atlanta, Georgia, AIAA Paper 88-4622.

Hosman, R.J.A.W. (1996). Pilot's perception and control of aircraft motions. Ph.D. thesis, Delft University of Technology. Delft University Press, Delft, the Netherlands. ISBN 90-407-1384-7.

Hosman, Ruud J.A.W. (1997). Visual-Vestibular Interactions in the Perception and Control of Aircraft Motions by the Pilot. AIAA Modelling and Simulation Technologies Conference. New Orleans, LA, August 11-13, 1997. AIAA CP-97-3802.

Hosman, Ruud and Henk Stassen (1998). Pilot's Perception and Control of Aircraft Motions. IFAC/IFIP/IFORS/IEA Conference on: ANALYSIS, DESIGN AND EVALUATION OF MAN-MACHINE SYSTEMS. KYOTO, Sept.16-19, 1998.

Junker, A.M., and C.R. Replogle (1975). Motion Effects On the Human Operator in a Roll-Axis Tracking Task. Aviation, Space and Environmental Medicine, 46: 819-822.

Kingston, P., T. Peters, C. Lehman, and E.M. Boothe (1996). The Real Training Value of FTD's in an Airline Training Program. Web side: www.Atlantis.com.

Levison, W.H., and A.M Junker (1978). A model for the Pilot's Use of Motion Cues in Steady-Sate Roll-Axis Tracking Tasks. AIAA Flight Simulation Technology Conference. Arlington, Texas

Mesland, B.A. (1998). About Horizontal Self-Motion Perception. Ph.D. Thesis University of Utrecht, The Netherlands.

Meiry, J.L., (1965). The Vestibular System and Human Dynamic Space Orientation. Massachusetts Institute of Technology, Man-Vehicle Lab. Rept.T65-1, Sc.D. Thesis./ NASA CR-628, 1966.

Newell, F.D. (1968). Human Transfer Characteristics in Flight and Ground Simulation for the tracking Task. Wright Patterson AFB, Ohio, AFFDL-TR-67-30.

Nahon, M.A. and L.D. Reid (1990). Simulator Motion-Drive Algorithms: A Designer's Perspective. AIAA Journal of Guidance, VOL.13, NO.2, pp 356-362.

Ray, Paul A. (1996). Quality Flight Simulation Cueing. Why? Proceedings of the AIAA Modeling and Simulation Technology Conference. AIAA-96-3488-CP.

Ray, Paul A. (1999). Quality Flight Simulation – Proper and Improper Applications. Proceedings of the Symposium on "Can Flight Simulation Do Every Thing?". Royal Aeronautical Society, London, May 19 and 20, 1999.

Rasmussen, J. (1983). Skills, Rules, and Knowledge; Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models. IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No. 3, May/June 1983.

Stapleford, R.L., R.A. Peters, and F.R. Alex, (1969). Experiments and a model for pilot dynamics with visual and motion inputs. NASA CR-1325.

Steen, F.A.M. van der (1998a). Self-Motion Perception. Ph.D. Thesis. Delft University of Technology. Delft, The Netherlands. ISBN 90-9011757-1.

Steen, F.A.M. van der (1998b). An Earth-Stationary Perceived Visual Scene During Roll and Yaw Motions in a Flight Simulator. Journal of Vestibular Research, Vol. 8, No. 6, pp 411-425, 1998.

Teunissen, R.P.A.M. (1997). Ab-initio ZFT: Creating Pilots without wings. Proceedings of theConference "Flight Simulation - Expanding the Boundaries". Royal Aeronautical Society, London, May, 15-16, 1997.

ANALYTICAL EVALUATION OF A SIMPLE PARTIAL GRAVITY BALANCING METHOD FOR LIMITED-POWER FLIGHT SIMULATION

Moshe Idan* and Arthur J. Grunwald†
Department of Aerospace Engineering
Technion, Israel Institute of Technology, Haifa 32000, Israel

The main disadvantage of limited-power and low-cost electrical simulator drive systems is that continuous power is required to balance the static gravity forces. This reduces the power available for the dynamic actuation of the simulator and effectively reduces the bandwidth and range of the motion base. In this paper, a parallel mechanism is presented for partial static gravity balancing, thus channeling most of the available electrical power to the dynamic actuation of the simulator. The addition of a pneumatically operated passive mechanism is proposed, placed as a "seventh leg" under a six-degrees-of-freedom Stewart platform, i.e. a hexapod setup with six electrically actuated legs. A numerical simulation was carried out to evaluate the effectiveness of the support mechanism. Both for maneuvers demanding slow, long-stroke motions of the actuators, as well as for fast small-amplitude sinusoidal excitation, the support system yielded electrical power savings of over 70%. For high-frequency sinusoidal excitation, at least 35% larger amplitudes were possible. An electrical drive system, fitted with the proposed passive pneumatic support unit, is expected to be an attractive alternative solution for obtaining flight simulator fidelity at moderately increased cost.

Introduction

RECENT progressive developments in ground and air transportation has largely increased the complexity of the aircraft piloting or road vehicle driving task. The advent of modern electronics and display systems has facilitated the development of low-cost research and training tools, used for studying the interaction of the human controller, or preparing him/her for dealing with this complex task. Moreover, flight trainers or driving simulators allow the pilot or driver to obtain hands-on experience in a safe laboratory environment.

A crucial component of these simulation systems is a high-fidelity motion platform. Commercial flight simulators, i.e. as used by the airline companies for flight crew training, commonly employ a Stewart Platform, consisting of six hydraulically-driven linear actuators. The overall cost of these simulators may be well over 20 million dollars and the platform payload can be up to 4 tons of mass[1].

Our proposed development addresses the low-cost trainers, with a platform payload of less than 400kg and an overall cost of under $200,000. The research flight simulator, developed and build at the Philadelphia Flight Control laboratory at the Technion - IIT[2], is a unique development in which the six legs of the Stewart Platform are actuated by an electrically driven system. Since electrical actuation systems are one order of magnitude lower in cost than hydraulic actuation systems, six-degrees-of-freedom motion simulation is achieved at a fraction of the cost compared to a commonly used hydraulic platform.

However, the main disadvantage of such a limited-power, low cost electrical simulator drive system is the need for continuous power to balance the static gravity forces. This need does not exist in hydraulic systems, since here the weight is balanced by the pressure of the incompressible fluid trapped in the actuator. The disadvantage of the continuous power requirement is two-fold: (1) it reduces the power available for the dynamic actuation of the simulator, effectively reducing its dynamic range and limiting the bandwidth, and (2) it causes increased wear-and-tear in the mechanical drive system itself, demanding more frequent servicing and overhaul.

The study presented in this paper examines a simple pneumatic passive support mechanism, for partial static gravity balancing, that would nearly provide the required continuous gravity balancing force. Consequently, the available electrical power,

* Senior Lecturer, Associate Fellow AIAA
† Associate Professor
Copyright ©1999 by Moshe Idan and Arthur J. Grunwald. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

previously used for the gravity balancing, would be channeled to the dynamic actuation, improving the dynamic characteristics of the simulator, in addition to a reduction in wear-and-tear. The electrical drive system, fitted with such a passive support unit, would be a commercially attractive solution, providing low-maintenance high-fidelity flight simulation for a fraction of the cost of its hydraulic alternative.

Passive Support Mechanism

An ideal static support mechanism would exert supporting forces on the simulator structure, such that the resultant of these forces would be vertical in upward direction, pass through the center of mass and be of a magnitude equal to the static weight of the simulator, for any angular orientation and position of the platform. Since the platform moves in all six degrees-of-freedom, the supporting forces needed to achieve ideal static balancing will change in magnitude and direction with respect to the moving platform, and thus can be realized only with complex mechanisms[3]. Partial balancing refers to an non-ideal support mechanism. With such a mechanism, having a simplified geometry, simulator motions might result in: (1) incomplete static weight cancellation; (2) residual moments and forces.

Passive gravity balancing mechanisms, acting in parallel with the drive system, have been considered in literature[4]. Passive support refers to an actuator, working in parallel with the servo-controlled electromechanical drive system, of which the forces are not actively controlled and through which no energy is added to the system.

The most obvious solution for counterbalancing the gravity forces would be suspension of the platform by counter weights. However, this method would double the mass of the simulator system, strongly reducing its bandwidth and thus limiting the high-frequency performance of the simulator. It would also need a system of levers or pulleys and demand a volume of space for the counter weights, making this method unpractical.

A second candidate for counterbalancing the simulator weight would be to use a system of mechanical springs and masses. The disadvantage of springs is that they are difficult to adjust to changing payloads, and that the combined simulator spring-mass system might cause undamped oscillations. Ideal gravity balancing using such systems, also leads to complex and expensive mechanisms[3].

In this study, a pneumatic suspension system is proposed for partial gravity balancing of dynamic motion-base flight simulators. The pneumatic system allows easy adjustment to changing payloads, provides supporting forces that are almost independent of the deflection, and provides a well-damped system.

The suggested simple pneumatic support system is shown schematically in Fig. 1. The air cylinder is connected to a large pressure vessel through an open pipe. The pressure in the vessel is adjusted such, that when the platform is at rest in the nominal position, its static weight is canceled. As the platform starts moving, the pressure in the cylinder will initially change. These pressure changes will cause the air to flow to or from the vessel. Since the volume of the vessel is very large in comparison to the volume of the cylinder, the pressure in the vessel will remain almost unchanged. Due to the open connection between the cylinder and the vessel, the pressure will equalize after the change has taken place, and the supporting force will resume its nominal value. However, due to a fixed connection of the support system to the ground and to the platform, motion of the latter will introduce residual (non-balancing) forces and moment. The ultimate design goal of the final support system will be to optimize its geometry and parameters to minimize this effect.

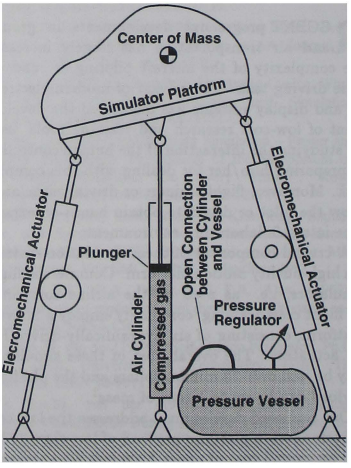


Figure 1: Schematic representation of the pneumatic support system using an air cylinder connected to a large pressure vessel.

Support System Model and Analysis

To evaluate the effectiveness of the above support system, preliminary paper simulations were carried out. A full-scale six-degrees-of-freedom electrically driven motion-base flight simulator model, validated experimentally in Ref. 5, was used in this simulation. The motion platform was augmented with the proposed simple pneumatic support unit. For simplicity, this pneumatic leg is supporting the platform at a position directly under the center of mass of the simulator cabin. The pressure in the support system is tuned initially to fully balance the simulator weight at its nominal static position.

The dynamic action of the pneumatic system is shown in Fig. 2. The length of the cylinder is l, and the pressure and the volume of the gas contained in it are P and V, respectively. For a platform at rest, P and V are at their nominal values P_0 and V_0, respectively, where P_0 is the pressure in the gas vessel. As the platform starts moving, $P = P_0 + p$ and $V = V_0 + v$, where p and v represent small changes in the pressure and volume, as a result of the motion. Since the vessel volume is large compared to the volume of the gas contained in the cylinder, it is assumed that the pressure in the vessel remains constant during the cylinder motion.

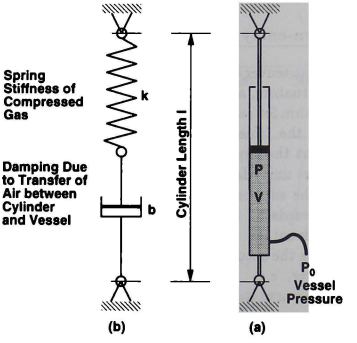


Figure 2: Dynamic spring-damper model of the pneumatic support system.

The equation of gas in the cylinder is given by

$$P = \rho RT \tag{1}$$

where $\rho = M/V$ is the gas density, and M is the mass of the air contained in the volume V. $R = 287.06[\mathrm{m}^2/\mathrm{sec}^2 \cdot {}^\circ \mathrm{K}]$ is the specific gas constant of air, and T is its temperature in degrees Kelvin. Assuming the temperature of the gas is constant, substituting $\rho = M/V$ in Eq. (1) yields

$$\frac{PV}{M} = \frac{P_0 V_0}{M_0} = \text{Constant} \tag{2}$$

Therefore, M, P and V are related by

$$M = C_a PV \tag{3}$$

where $C_a \triangleq \frac{M_0}{P_0 V_0}$.

For small changes p and v, the small change in the mass m follows from linearizing Eq. (3):

$$m = C_a(V_0 p + P_0 v) \tag{4}$$

Assuming an outward mass flow to be proportional to the pressure increase p, the rate at which M increases is given by:

$$\dot{m} = -C_m p \tag{5}$$

Differentiating Eq. (4) and using Eq. (5) yields

$$C_a(V_0 \dot{p} + P_0 \dot{v}) = -C_m p \tag{6}$$

The volume rate of change is given by

$$\dot{v} = A\dot{l} \tag{7}$$

where A is the cross area of the cylinder. The change in the support force of the support system is expressed by

$$\Delta F = Ap \tag{8}$$

Using Eqs. (7) and (8) yields, after Laplace transformation, the transfer function between ΔF and $\dot{l}$

$$\frac{\Delta F(s)}{\dot{l}(s)} = -\frac{A^2 P_0/V_0}{s + C_m/(C_a V_0)} \tag{9}$$

Eq. (9) can be represented by an equivalent spring-damper system shown in Fig. 2. It can be easily shown that

$$\frac{\Delta F(s)}{\dot{l}(s)} = -\frac{k}{s + k/b} \tag{10}$$

where k symbolizes the "spring stiffness" of the compressed gas in the cylinder, and b represents the damping effect, or the energy loss in the transfer of the air from the cylinder to the vessel and versa. Comparing Eqs. (9) and (10) yields

$$k \triangleq \frac{A^2 P_0}{V_0} \left[\frac{\mathrm{N}}{\mathrm{m}}\right] \tag{11}$$

and

$$b \triangleq A^2 P_0 \frac{C_a}{C_m} = \rho_0 \frac{A^2}{C_m} \left[\frac{\mathrm{N}}{\mathrm{m/sec}}\right] \tag{12}$$

In order to minimize the force fluctuations ΔF due to the spring-damper effect of the support system, k and b should be kept as small as possible. Eqs. (11) and (12) show that both k and b reduce with a reduction in the cross section A. Eqs. (11) also shows that the spring stiffness k decreases with an increase in the volume V_0 of the gas contained in the cylinder. A smaller cross section A imposes the requirement for a higher vessel pressure P_0, needed to obtain the same supporting force for a given simulator cabin weight. Thus k can be reduced by increasing the cylinder volume V_0, by increasing the gas pressure P_0 and, as a result of the latter, by reducing the cross section A. Eq. (12) shows that C_m should be large, which means that the restriction of the connection with the pressure vessel should be as small as possible, i.e., the connecting pipe would be short and wide.

Possible solutions for minimizing k and b by keeping A small and V_0 large are presented in Fig. 3. Solution (a), which uses a long cylinder, might require the creation of a space under the simulator floor; solution (b) uses an enlargement at the lower part of the cylinder and solution (c) a "double" concentric set of cylinders.

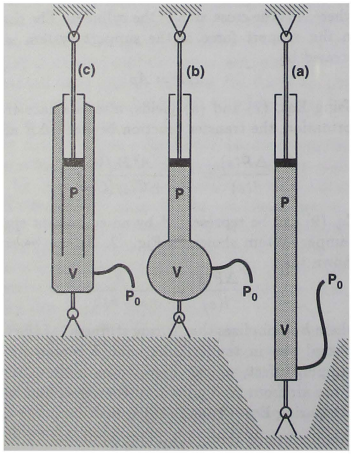


Figure 3: Solutions for reducing the plunger cross section A and enlarging the cylinder volume V.

Numerical Tests and Results

The analytical model of the full-scale six-degrees-of-freedom electrically driven motion-base flight simulator, augmented with the model of the proposed simple pneumatic support unit, was used in numerical simulations to evaluate the performance of the new system. To support the static weight of the simulator cabin, the nominal supporting force was set to about 4000N, k was set to 100N/cm and b was chosen to be 5N/(cm/s).

The advantages of the suggested support system are analyzed by evaluating the savings in the electrical power required to move the simulator, the reduction in the actuation forces, and the increase in the high frequency dynamic range of the simulator. All these factors can be utilized for improving the dynamic performance of the flight simulator and for increasing the bandwidth of the drive system. The reduction in the actuation forces may directly contribute to reducing the wear-and-tear of the simulator, increasing the mean-time-between failures, and reducing the need for frequent overhauls.

Two basic maneuvers (motions) of the simulator were considered: (1) simulation of a turn-entry maneuver of a B-747 aircraft, and (2) vertical sinusoidal motion of the cabin, which can simulate turbulence or vertical vibrations of a rotorcraft.

A turn-entry maneuver of a B-747 aircraft

This maneuver demands slow, long-stroke motions of the actuators, as commanded by a classical washout algorithm for realistic motion cue generation[6]. Fig. 4 shows the time-histories of the six actuator forces without the support unit (solid lines) and with the support unit (dashed lines). The graphs clearly show that the support system generally shifts the curves downwards by about 800N. For actuators 1 and 2, the advantage of this shift is the largest, since it reduces the actuator force from 800 N to almost zero. However, e.g. for actuators 5 and 6 at about 20 sec, the actuator force without the support system is almost zero, whereas with the support system this force is about -500N. This negative force is caused by the fact that for this situation the platform is rolled strongly. Since the center of mass is well above the platform floor, the actuators on the opposite side to the roll need to exert a pulling force to prevent the platform from toppling over. The same is true for actuators 3 and 4 at about 5 and 30 sec. The overall electrical power saving as a result of the support system, averaged over the entire maneuver, is 85%.

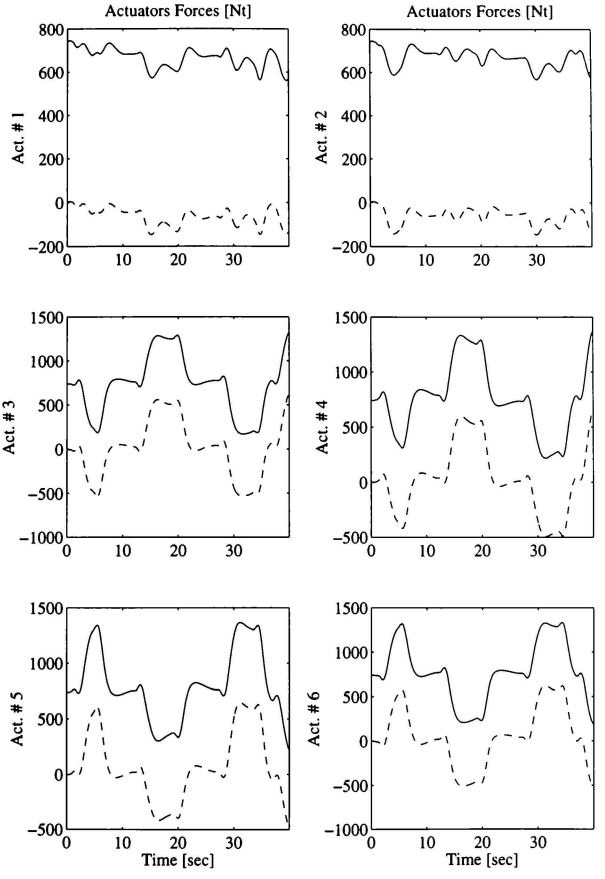


Figure 4: Time-histories of the six actuator forces for the turn-entry maneuver: solid lines - without the support unit; dotted lines - with the support unit.

Sinusoidal vertical motion

Vertical turbulence can be represented by sinusoidal small-amplitude heave excitation of the simulator. The upper graph in Fig. 5 shows the average electrical power used in these simulator motions as a function of the excitation frequency. The graph shows that the support system shifts the curve of the electrical power downwards by about 400 Watt. The lower graph shows the savings in percentage. Obviously the savings are 100% for the static case and decrease with the frequency of excitation. Typical aircraft turbulence or buffeting would be in the frequency range of 10-20 rad/sec, for which the power savings are well above 75%.

High frequency motion amplitude of a limited power flight simulator is strongly limited by the power or current limitations of the drive system. The sinusoidal vertical motion experiments were used to evaluate the maximum amplitude which could be achieved with the simulator without reaching the power limit. Figure 6 clearly demonstrates that the maximum amplitude increases by 30-35% when using the suggested support system. This increase is attributed to the lower power and thus lower current required by the drive system to perform a specific motion in the presence of the support system.

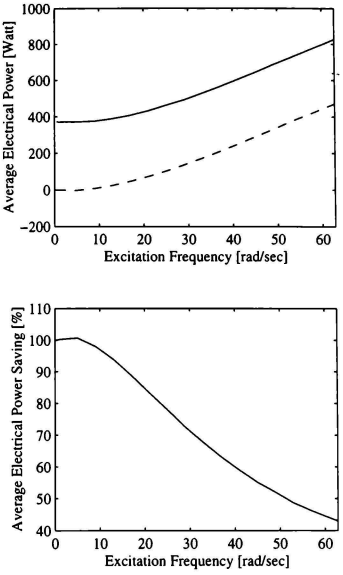


Figure 5: Average electrical power (upper graph) and power saving (lower graph) as a function of excitation frequency. Upper graph: solid line - without the support unit; dashed line - with the support unit.

Discussion

The above examples show that both for slow, large-stroke motions, as experienced in maneuvering aircraft as well as for the simulation of turbulence, the static support system could yield power savings of well over 70%, which clearly demonstrates the potential benefits of the suggested method. In addition, the increased high frequency motion amplitude could be beneficial in simulating such phenomenon as buffeting and turbulence.

Conclusions

The large electrical power savings resulting from the "seventh leg" support mechanism demonstrate that even simple partial balancing solutions might be very beneficial. More complex passive mechanism might result in more complete balancing over a wide range of operating conditions. Whether the added complexity is justified, depends on the use of the simulator. If the supporting force of the "seventh leg" support mechanism is vertical and passing through the center of mass of the simulator in its nominal operating position, the static gravity forces can be balanced entirely. As the platform deviates from its nominal operating position, the support force might no longer be vertical and/or pass through the center of mass of the simulator. Hence, the effectiveness of the single support force decreases with the platform deviations from its nominal state. Simulators, designed for maneuvers of which the deviations about the nominal state are small, i.e. simulating buffeting or turbulence, might highly benefit from the simplicity of the single-leg support. However, simulators that require slow and long-stroke maneuvers might deviate to such a degree, that the single-leg support is no longer effective. In this case a more complex support mechanism is required, the kinematics of which should be designed such that even for large deviations, the resultant of the support forces is close to vertical and passes through the center of mass. Hence, in finding a trade-off between support mechanism complexity and balancing

quality, the characteristics of the required simulator maneuver play a key role.

Acknowledgment

This research was partially supported by the fund for the promotion of research at the Technion.

References

1. Advani, S.K., *Simona - A reconfigurable and versatile research facility*, AIAA Modeling and Simulation Technologies Conference, New Orleans, August 1997, AIAA-97-3809-CP.

2. Gandelman, G., *Design of a control system for a six-degrees-of-freedom dynamic simulator*, M.Sc. Thesis, Faculty of Aeronautical Engineering, Technion - IIT, Nov. 1987.

3. Leblond, M. and Gosselin, C.M., *Static balancing of spatial and planar parallel manipulators with prismatic actuators*, Proceedings of DETC-98, ASME Design Engineering Technical Conference, Sep. 13-16, 1998, Atlanta, Georgia, USA.

4. Ulrich, N. and Kumar, V., *Passive mechanical gravity compensation for robot manipulators*, Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, California, USA, pp. 1536-1541, 1991.

5. Idan, M. and Sahar, D., *A robust controller for a dynamic six-degrees-of-freedom flight simulator*, Proceedings of the AIAA Flight Simulation Technologies Conference, San Diego, California, pp. 53-60, July, 1996.

6. Nahon, M.A. and Reid, L.D., *Simulator motion drive algorithms: A Designer's Perspective*, AIAA Journal of Guidance, Control and Dynamics, 13(2):356-362, Mar.-Apr., 1990.

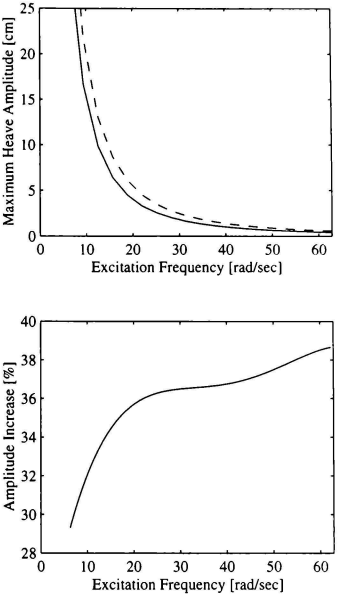


Figure 6: Maximum heave motion amplitude (upper graph) and amplitude increase (lower graph) as a function of excitation frequency. Upper graph: solid line - without the support unit; dashed line - with the support unit.

COLLABORATIVE DECISION MAKING AIRSPACE VISUALIZATION TOOL

Jimmy Krozel, Ph.D.,
David Schleicher, and Darren Dow
Seagull Technology, Inc.
16400 Lark Ave.Los Gatos, CA 95032-2547

Abstract

This effort investigates the design of a Collaborative Decision Making (CDM) visualization tool to aid dispatchers at Airline Operations Centers (AOCs) collaborating with Air Traffic Management (ATM) in the National Airspace System (NAS). As new *Free Flight* procedures remove jetway routing, positive control, and other constraints, an added emphasis will be placed on collaborative ATM techniques and distributed control. Collaboration can occur through telephone conversations between two users both working with a common visualization tool and using shared artifacts including different color pointers for each user, notepads and audio notes for message passing, and other shared constraint objects in the visual display. This tool can assist in keeping track of aircraft, observing weather systems and special use airspace constraints, and viewing operations data like gate assignments and ground holds.

Introduction

This effort investigates the design of a CDM visualization tool to benefit dispatchers at AOCs, aircraft flight crews, and traffic flow managers for ATM, which form a collaborative triad as shown in Figure 1. Compared to today's methods of ATM, new *Free Flight*[25] procedures will allow for a greater emphasis on collaborative ATM techniques and distributed control. A number of Free Flight applications can benefit from this visualization tool; these include, but are not limited to:

- Collaborative Routing (around weather, congestion, etc.)
- Collaborative Departure Scheduling
- Collaborative Arrival Planning, and
- Post- or Real-Time- Operations Feedback.

Our tool allows for monitoring information and collaborating with a common visual model. Our emphasis is on the AOC-ATM link of the triad. This visualization tool will assist NASA, the FAA, and airspace users in achieving effective and safe control of multiple aircraft in the NAS through the visual integration of air and ground-based air traffic information.

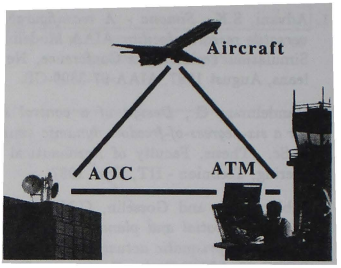


Figure 1. The airspace visualization tool is applicable to AOC-ATM tasks within a triad system formed by the Airlines Operations Center (AOC), the aircraft flight crew, and Air Traffic Management (ATM).

This collaboration airspace visualization tool can be used to assist in a number of problems:

- monitor and discuss flights, traffic congestion, and aircraft separations,
- observe weather conditions around the country,
- observe ATM sector transitions,
- observe ground and airborne delays and react to them,
- view potential conflicts under Free Flight conditions as well as today's conditions, and display, compare, negotiate, and select conflict resolution solutions,
- visualize constraints like Special Use Airspace (SUA), Flow Constrained Airspace (FCA), SIGMETs, turbulence avoidance regions, and discuss alternative solutions abiding by these constraints, and
- facilitate solutions to traffic flow management problems through collaborative routing.

The CDM tool facilitates two or more users collaboratively solving problems related to the efficient use of the NAS; the nominal architecture follows the concept illustrated in Figure 2.

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights Reserved.

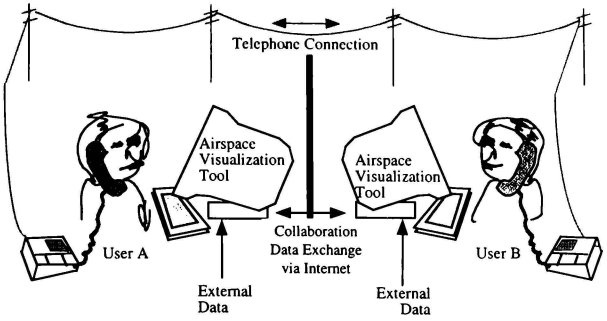


Figure 2. A 2-user application of the collaboration system.

Two users can use the collaborative decision making tool to communicate, or they may augment their communication by a concurrent telephone conversation. At the fingertips of each user is the 3D visualization system, driven by streams of data including aircraft positions, flight plans, weather, pilot reports, gate assignments, delay information, etc. Ideally, each user has the same input stream of data, so that none of these data needs to be exchanged from one computer to the next during a collaboration session. Also, these data being identical will result in the same 3D virtual world being observed by each user. The information that is exchanged during a collaboration session is minimal. This information may include the user's cursor (pointer) location during the collaboration, text and audio messages passed back and forth, and data that present the constraints (e.g., FCA) or solutions (e.g., a new flight plan) to problems solved by one user to be discussed with a collaborator.

Related Work

The display of our visualization system is being designed as a 3D perspective Virtual Reality (VR) type display. VR is a computer-generated representation of 3D space that enables a user to experience immersive, interactive simulation of either a realistic or imaginary environment. A VR system works by creating the sensation that the user is navigating though and interacting with a 3D world, even though the 3D world is being displayed on a 2D computer display. Our VR environment does not have extremely high fidelity, but will allow the user to recognize important objects (e.g., aircraft, airports, and weather), navigate through a 3D space, and collaborate with users. Some tools for collaboration in VR environments and some related airspace visualization displays are discussed next.

Collaboration tools in VR have been developed in two primary frameworks. First, collaboration has been researched for users at geographically remote locations (like the application in this paper) and, second for users in a common shared physical space (e.g., the same room) but collaborating together in a virtual world. The work performed at the Human Interface Technology Lab of the University of Washington focuses on the use of shared spaces in virtual worlds[4]; the shared space allows two (or more) users to work in collaboration in the same shared physical space and share virtual objects with one another. In Poupyrev *et al*[23], virtual notepads with hand written messages are used in such a shared space. In a virtual scientific study room[14], the system allows for collaborators to visualize and work together studying 3D scientific data. Head mounted displays and head trackers are used to determine the location of the 3D virtual images to be displayed to each collaborator. In another example, Georgia Tech has been producing software tools to assist in collaboration for VR environments[7]. Their pen-based metaphor allows the user to control a pen object in the VR environment; a pen and tablet can be used in the virtual environment or the pen can be used as a pointing device (with a laser pointer tip). Their audio annotation technology allows audio messages to be read to the user based on user click selection, based on the user passing through a plane within the virtual environment, or based on coming within a certain range from an object.

The Collaborative Virtual Environment (COVEN) is a collaboration system being developed by several European industrial and academic constituents addressing the technical issues behind VR-based multi-user collaboration[20]. COVEN is exploring direct-manipulation and speech-based multi-modal interaction, trade-offs between realism and level of detail, real-time animated (human) bodies, facial expressions, collective transportation — guided navigation — for a group of users, and network scalability issues for many users. COVEN exploits a Distributed Interactive Virtual Environment (DIVE), a process of exchanging peer-to-peer communications between virtual worlds[10].

Some of the latest display techniques applied to the air traffic visualization are presented in conference and journal articles[1,3,8,9,24]. Most of the recent display issues have been focused on the benefit of 3D or perspective type displays, and object-oriented text[1,11] or scene-linked symbology[19]. Smith, *et al*[27] identify that when a display does not show the third spatial dimension as readily apparent as the other two, pilots tend to solve problems, in this case conflict avoidance problems, in the displayed two directions more often than in the three dimensions. As a follow-on to this study, a perspective display was provided to pilots for traffic avoidance, and the pilots were more likely to choose a solution with a vertical component[13]. This work motivated us to investigate a 3D perspective display for this research.

Collaborative Decision Making in the NAS

The idea of collaborative problem solving among the major active agents in the NAS such as AOC dispatchers, AOC coordinators, traffic flow managers, air traffic controllers, and flight crews has been gaining momentum in the past decade. Historically, each of these entities have been operating according to their legal roles described in the Federal Air Regulations (FARs) with their own separate data sources, displays, and decision support systems. However, past experience by these agents have identified numerous situations where they end performing actions that end up counter-productive to one another.

One example of this problem occurs with the operation of the FAA's Ground Delay Program (GDP). When circumstances such as adverse weather reduce the capacity of a major US airport, FAA traffic flow managers will issue a GDP that controls the departure times of flights still on the ground that are destined for the impacted airport. Among the problems faced by GDP actions is the fact that FAA flight information is based on OAG schedules, which will not reflect recent airline schedule changes. The airline, who may expect airport acceptance rates to go down, often make significant changes to their schedules (including canceling flights) which tend to render the data that the FAA traffic flow managers are using to be inaccurate. Why don't airlines provide the FAA with these updated schedule information? According to current procedures, if an airline tells the FAA of a canceled flight, not only will the FAA take the flight out of its GDP, but it will not allow the airline to fill this "slot" previously occupied by the canceled flight with another aircraft. Additionally, if an airline report a delay to ATM, ATM will re-project the flight's arrival time. If a GDP involves this flight, an additional delay to the reported delay is incurred – a "double-penalty". Recent FAA programs (e.g., Ground Delay Substitutions Visualizer, FAA/Airline Data Exchange, and CDM) have sequentially targeted the development of new GDP procedures, data and decision support systems that remove this "double-penalty" issue and foster solutions that result in collaborative airline and FAA problem solving which decrease the need and impact of GDPs[35].

A significant number of research projects have recently been performed to find ATM solutions involving collaboration and supporting decision support systems amongst NAS agents[18,21,22,26,28,29,30,34]. Concepts in these papers include: control by permission, distributed problem solving, dynamic information transfer, and collaborative routing. These research efforts illustrate how the FAA, airlines, and NASA are responding to an increasing government and aviation industry interest in improving NAS capacity, efficiency, and flexibility in light of expected future air travel demand for NAS resources. A combination of emerging collaborative ATM decision-making concepts and significant economic benefits to the airlines[2] make for fertile ground for new ATM and AOC decision support tools and displays like the one investigated in this paper.

In our detailed discussions with dispatchers about the future needs of the airline in terms of decision support tool and visualization capabilities, a number of future airline needs and desires were identified:

1) The dispatchers felt that the recent explosion of relevant decision support tool capabilities should be integrated, especially from the visualization standpoint.
2) The dispatchers would like to know the airspace constraints that the FAA's ATM personnel are facing and to receive proactive information

regarding planned traffic flow management actions. Also, the ability to reliably project these airspace constraints into the future was desirable.

3) Given the knowledge of the FAA constraints, the dispatchers would like to have the tools to help identify the impact that such constraints will have on their operations.

4) Ultimately, the dispatchers would like to have the tools to help evaluate the potential impact of different airline intent choices to the FAA constraints on their operations.

Potential Benefits

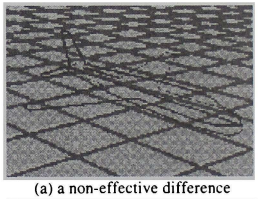
A 3D airspace visualization tool can support airline-ATM collaboration through better audio and visual communications by the ATC Coordinator, Dispatchers, ATC System Command Center, and Traffic Management Units. These visualization system interfaces should be able to provide an integrated set of airline-provided and ATM-provided data and voice information through their human-centered displays. The advantages of this new approach over the existing one include:

- A unified ATM and airline situation awareness of weather, airspace, aircraft and TFM intent, and routing constraints through the use of common data sources, common object representations (models), and an explicit visualization of ATM constraints, TFM intent, and airline intent. The visualization of ATM constraints should assist in improved airline flight plans that proactively avoid routing problems and improved airline routing requests. The ATM visualization of weather and airline preferences (which will have imbedded knowledge of important information such as remaining fuel on board) will enable the incorporation of additional safety factors into the TFM strategy;
- Higher dispatch and traffic flow management productivity through the integration of the visualization systems and existing airline-specific or ATM-specific tools;
- A collaborative approach to solving routing-related problems that includes regular and advanced warning of planned TFM actions that allow the airlines time to proactively react, allow for significant input from the airlines, and circulate ATM-provided reasons for disapprovals to airline routing suggestions;
- Improved re-routing constraint and intent dissemination to airspace users through the use of alerts and non-time-critical voice annotation information; and
- Short time lags and reduced workload requirements based on the use of a computer-based representation of routing requests, routing constraints, and routing approval/disapproval.

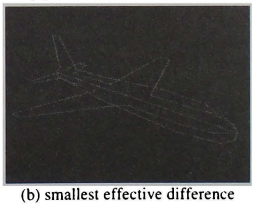
The benefits to this future Collaborative Routing concept include not only higher dispatcher and traffic flow management productivity, but also less disruptive and less-hazardous re-routes.

Design

Our CDM visualization tool is being designed using human-centered automation design guidelines, inspired by the work of Billings[5,6]. For instance, we are designing the tool for operators to be informed while being involved, with straight-forward monitoring, with predictable mechanics, and communication of intent. Furthermore, for the visualization tool options and user interfacing, design methodologies espoused by Tufte[31,32,33] are being followed. For instance, we are using layering and separation, micro and macro readings, establishing continuity in space and time for good situational awareness, employing the smallest effective difference principle when possible, and using standard encodings easily recognizable by all users groups. Figures 3 and 4 illustrate a couple of these concepts.

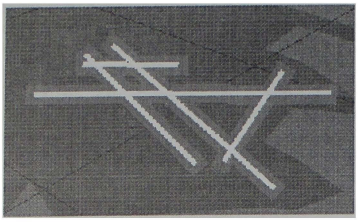


(a) a non-effective difference

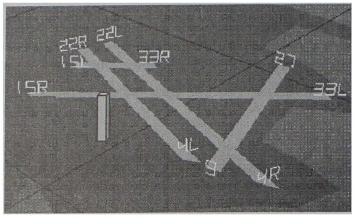


(b) smallest effective difference

Figure 3. The smallest effective difference can de-emphasize secondary visual items, like grid lines, in comparison to primary visual items, like aircraft.



(a) symbolic model



(b) representational model

Figure 4. Multiple object models of an airport are used based on the micro/macro reading of this object; (a) if far away, a symbolic model is used (note: when the airport symbol is colored blue, then the airport has a control tower), and (b) if closer, a representational model is used (runways and control tower are modeled in 3D perspective).

Standard encodings are used within the CDM system for weather, navigation data, and other data. For instance, weather radar displays most typically use green for light rain (30-39 dBZ), yellow for heavy rain (40-49 dbZ), red for severe rain (50-59 dBZ) and magenta for extremely severe conditions (60+ dbZ). When displaying precipitation data in the visualization tool, these standards are used. Standard wind barb glyphs are also used, e.g., Light Winds ... Heavy Winds. For turbulence data, e.g., ▲, ⋀, and ⋀ are used. Figure 5 illustrates an example of weather data displayed in the visualization system. In this figure, wind, turbulence, and precipitation data are layered.

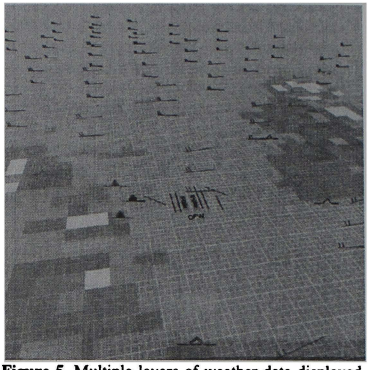


Figure 5. Multiple layers of weather data displayed concurrently (wind, precipitation, and turbulence).

As another example of standard encodings, navigation aids have standard symbols for VHF Omni-directional Range (VOR), ⬡, Tactical Air Navigation (TACAN), ▽, VOR/Tactical Air Navigation (VORTAC), ⬡, Distance Measuring Device (DME), □, and others. Waypoints are typically displayed as ✧. Airways are labeled on aeronautical charts as Victor Routes with common "V" numbers, *e.g.*, V123, as shown in Figure 6, and Jet Routes with common "J" numbers, *e.g.*, J123. Also from aeronautical charts, airports have standard symbology.

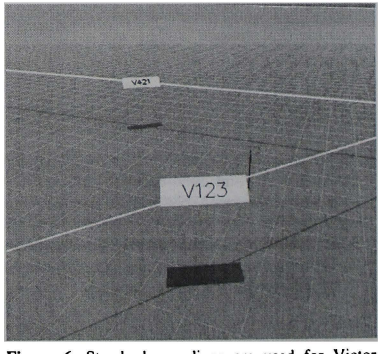


Figure 6. Standard encodings are used for Victor airways.

The final design philosophy is that the CDM system is being designed to be intuitive and interactive in the most natural method of display of information — a 3D graphics presentation. Gathering information is based on clicking on objects in the visualization world and selecting menus that create further 3D object results or provide further 2D displays. Navigating around an airport or navigating from airport to airport in the NAS is as simple as selecting airport choices and flying there. Navigation is controlled by a continuous motion governed by slewing functions[15]. Using slewing functions, we insure continuity in space and time in the visualization system, which facilitates good situation awareness. Thus, instead of instantaneously changing the viewers eye location, for instance, from one airport to another, the user is navigated quickly there. Navigation follows the transition function:

$$x(t) = x_{old} + c(t)(x_{new} - x_{old}) \qquad (1)$$

where the subscripts new and old denote the new position and old position. The slewing function c(t) follows a transition as shown in Figure 7. The best slewing functions rise and fall like a second order system with under-damping; discontinuities in the slewing function are found to adversely affect spatial awareness and, in some cases, to cause disorientation[15].

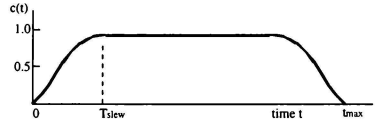


Figure 7. A slewing function c(t) that controls the transition from the old parameter setting x_{old} to a new parameter setting x_{new}, transitioning over the interval of time $t=0$ to $t=t_{max}$.

Communications Mechanisms for Collaboration

The primary tools for collaboration include the use of a telephone between collaborators while both viewing the 3D visualization system, focus of attention mechanisms, notepads, and audio notes. Preliminary designs for these collaboration tools are reviewed next.

Telephone Communications. Collaborators can discuss many problems with the use of telephone conversations while viewing the 3D scene from the CDM tool. However, the use of the telephone may be limited due to the fact that one ATM user may be servicing many airlines one after another, and might be limited to very short telephone conversations, if any.

Focus of Attention Mechanisms. Collaborators each have a pointer with a different color in the CDM tool. In this way, as one collaborator points out a feature, the other can see what is being highlighted. Additionally, any user can change the state of an aircraft, airport, or other feature in the visualization from static to blinking to facilitate focus of attention.

2D Notepads. Collaborators can leave messages for each other within the 3D scene using 2D notepads. Figure 8 illustrates an example of using a notepad to inform of a ground delay for DFW. Text messages can be typed in by a user and specified to be sent to one or more (or all) other users of the system. In this way, if a note is not intended for a specific user, the note never appears in the view of a user. One problem with 2D notepads is that long notes cannot be easily placed in small notepad areas, since the text becomes too small to read unless up close to the notepad. Thus, notepads are generally used for short messages which can be expanded into larger messages by "opening" the note, or these notepads can trigger audio messages or 2D graphical displays.

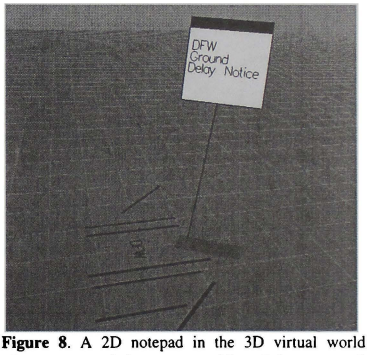


Figure 8. A 2D notepad in the 3D virtual world allows a user to inform one specific collaborator or all other users a short note; clicking on the note expands it to a full text message, audio message, or a 2D display.

2D Audio Notes. Collaborators can also leave messages for each other within the 3D scene using audio notes. Figure 9 illustrates an example of using an audio note to inform another user of a verbal message. One of the benefits of using audio messages is that the collaborator can be looking at

some other feature or can be moving around the 3D virtual world while listening to the message. Thus, the user does not have to have a fixed eye location so that text is legible (as is the case with text-based notepads). Audio notes can be used to augment a text notepad message or to replace a lengthy text message. Like with the other note mechanisms, audio notes can be specified to be sent to one or more (or all) other users of the system. In this way, if a note is not intended for a specific user, the note never appears in the view of a user.

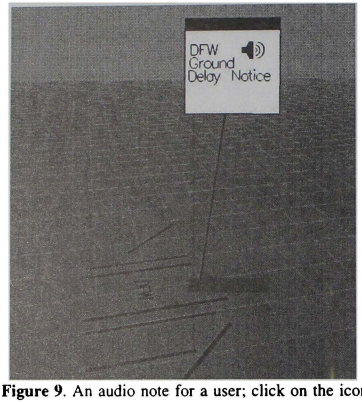


Figure 9. An audio note for a user; click on the icon and an audio message is played.

Audio notes can be input by a user, or pre-recorded audio messages can be used. In the case of pre-recorded messages, a user can select from a menu of pre-recorded messages and place the messages into the 3D scene through mouse input. Standard messages can relieve the user from having to take the time to record and playback a message to verify its correctness. An example is when a traffic flow manager establishes a Flow Constrained Airspace (FCA) for a given sector. The air traffic manager can place a text notepad and an audio note on the boundaries of the sector for all users to be informed of the FCA.

Interfacing to AOC/ATM Tools

The CDM tool is being designed to port to other systems that may reside in the AOCs or ATM. One of our objectives is to integrate one or more of the following tools into the CDM tool:

- Planning/Re-Planning software for fuel efficient 4D trajectory generation,
- Ramp Manager software for monitoring and changing gate assignments,
- Weather Avoidance software[16] designed for CTAS, the Center-TRACON Automation System, or
- Airline-specific software (supplied by an Airline).

This involves building input and output capabilities to operate tools like these. The objective is not to develop or redesign these tools. However, common to most tools is the ability to open a 2D window for input/output. The design of an efficient method for executing and displaying 2D windows within the 3D scene was developed for this purpose.

We are currently developing the CDM 3D visualization tool to be used by the airline dispatchers to monitor gate arrivals/departures/delays. In a recent effort to develop 2D display windows for this purpose, we have developed a "Psuedo-Ramp Manager" (PRM)[12]. The purpose of the PRM project was to emulate some of the tools found in the ramp tower of a major hub airline. The PRM employs a Gantt chart representation to depict the status of arrival banks expected over the course of a day. The PRM receives airline connectivity data for the aircraft, cabin crew, flight attendants, and passengers and displays these data to the user. The PRM is capable of supporting the following tasks:

1. monitor multiple arrival banks,
2. monitor and resolve problem alerts,
3. view information associated with flights,
4. update flight information for individual flights,
5. view gate characteristics,
6. modify gate characteristics,
7. make gate assignments,
8. find flights matching specific criteria, and
9. record data entered by the user during an application session.

Figure 10 depicts the main PRM 2D display from which the above tasks may be performed. Airport gates are shown on the left vertical axis, time is shown on the top horizontal axis and aircraft are shown relative to both gates and time. The Gantt chart bar size depicts the time interval an aircraft is expected to occupy a gate and its color conveys arrival/departure status and alert status. Table 1 lists the colors that were used and the status they signify.

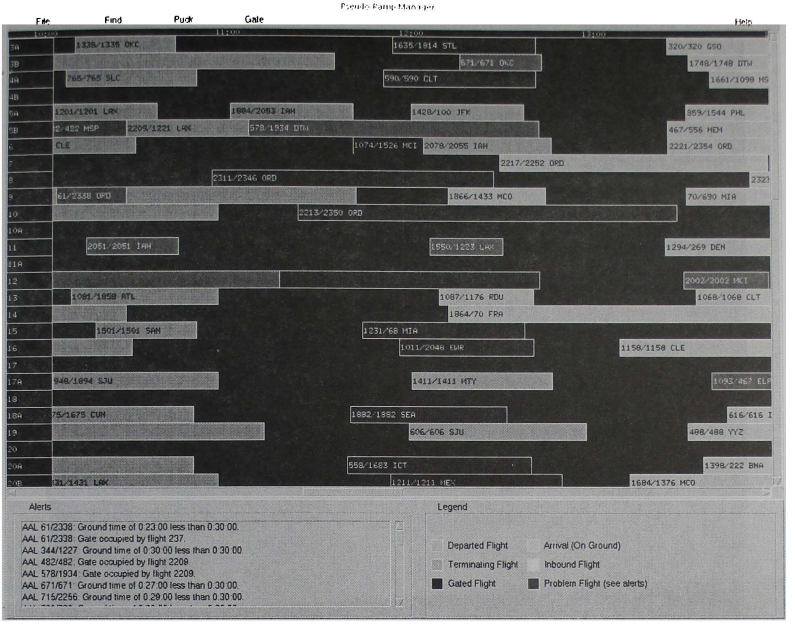


Figure 10. A gate assignment tool[12] for monitoring gate arrivals.

Table 1. Colors used to identify gate status.

Color	Status
Grey	Departed flight
Dark Grey	Terminating flight
Blue	Gated flight
Cyan	Arrived flight (but not gated)
Yellow	Inbound flight
Red	Problem flight

Gate information is accessible in the 3D visualization as a 2D display object within the 3D scene, as shown in Figure 11. This 2D display object is an intermediate display in that it does not allow full interaction with the gate assignment software. However, if the initial startup display provides sufficient information, there may not be a need to proceed further with the gate assignment software. For example, the 2D display object may indicate that an aircraft gate time window is shown in yellow and is on time; the gate assignment software does not have to be invoked. If the 2D display object indicates that the aircraft is not going to be able to meet a gate window of opportunity, a red bar is shown. Thus, the aircraft is late for the gate, and further action is necessary. If the gate assignment software is then invoked, a fully interactive 2D display window being controlled by the gate assignment software is opened. New 2D dialogue windows may also be opened by the gate assignment software, and when all gate assignment windows are closed and the application is exited, the 3D visualization software resumes control.

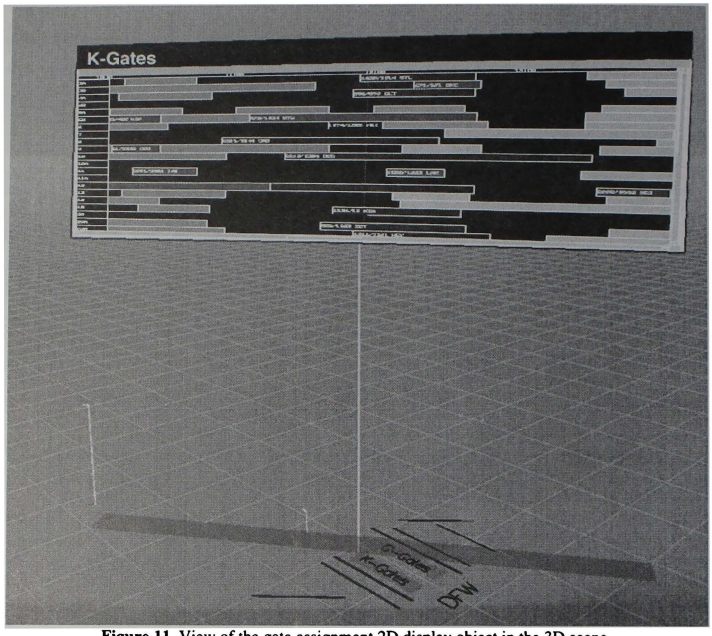


Figure 11. View of the gate assignment 2D display object in the 3D scene.

Demonstration Scenarios

The CDM 3D visualization system may be used by airline dispatchers to perform collaborative flight planning, flight following, and route re-planning around weather and other hazards. For a proof-of-concept software demonstration, a collaborative routing demonstration scenario is presented. This demonstration highlights how collaborative routing ATM-airline and intra-airline coordination and collaboration may be facilitated by the CDM airspace visualization tool.

Airline Problem Identification

First, as shown in Figure 12, the "Airline Problem Identification" demonstration illustrates how an AOC dispatcher may use the tool to spot a problem. A flight plan conflict with adverse weather has recently appeared due to adverse weather building up in en route airspace. The weather system is predicted to move slowly and possibly grow for the next hour or so. This adverse weather now presents a problem to a number of the airline's aircraft which have flight plans that go through the adverse weather region. The flight plan-weather conflict is spotted by an AOC dispatcher who wants to give notice to other dispatchers of the problem aircraft. Two responsible dispatchers refer to their visualization system to better understand the nature of these impending conflicts.

The visualization seen in Figure 12 is that being viewed by a dispatcher who is handling an active flight and wants to see the constraints posed by the upcoming weather conflict. In the visualization, the dispatcher can see the current position of his problem flight flying through en route airspace and its current 3D flight plan. In addition, the dispatcher has a view of the current adverse weather, other in-flight aircraft, jet routes, and state boundaries.

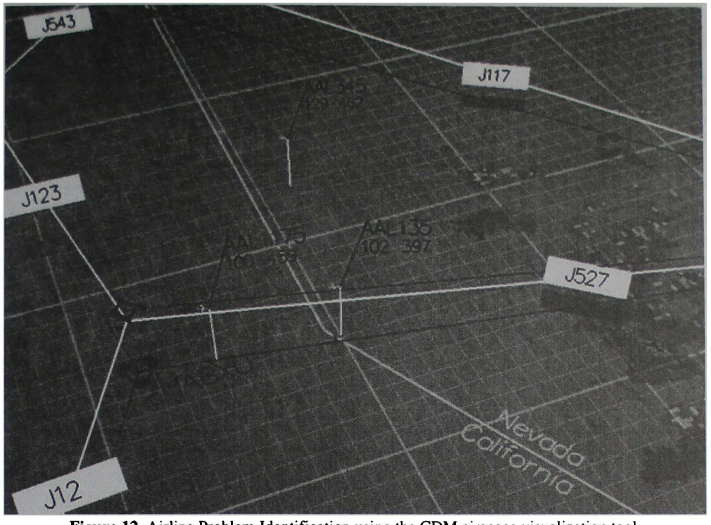


Figure 12. Airline Problem Identification using the CDM airspace visualization tool.

ATM-Airline Problem Coordination

Second, the "ATM-Airline Problem Coordination" demonstration shows aspects of the ATM-airline coordination process that takes place once ATM has identified the adverse weather problem. Air traffic controllers work to help vector aircraft to conflict-free routes around adverse weather. Simultaneously, traffic flow managers take steps to strategically react to weather constraints. ATM traffic flow managers study the weather pattern and the corresponding expected traffic-handling capacities of en route sector controllers. Based on predicted demand-capacity imbalances, traffic flow managers get together on the telephone and formulate a traffic strategy to react to the imbalance. Given tight capacity constraints caused by the weather persistence, traffic flow managers choose to define a Flow Constrained Airspace (FCA) around the en route sectors that contains the adverse weather. Also, traffic flow managers decide to direct upstream aircraft around the en route sectors until the weather attenuates.

Traffic flow managers input the FCA and its duration into the visualization tool to notify NAS users. Subsequently, this information shows up on the displays of CDM airspace visualization tool users. Because of the nature of the adverse weather, the traffic flow managers decide to avoid any ATM-airline teleconferences regarding the FCA. Instead, the traffic flow managers record an audio note describing the reasons for the FCA and distribute this information using the airspace visualization tool. Additionally, traffic flow managers use the visualization tool to identify all the aircraft that have flight plan conflicts with the FCA. Based on these data, the traffic flow managers send out email-based warnings to these airspace users requesting that aircraft with FCA-flight plan conflicts be re-routed around the FCA at least 30 minutes before arriving at the FCA.

Figure 13 illustrates the airline's ATC Coordinator's visualization tool after receiving the ATM information concerning the FCA and is currently working with a fellow dispatcher to create a re-route for the affected aircraft. Shown is a FCA volume, its expected duration, and an audio note that can be "double-clicked" to hear a recording of the reason for the FCA. Additionally, the current aircraft locations and weather are displayed. In planning an adequate re-route, the ATC Coordinator has enabled the visualization of navigational aids and jet routes.

Intra-Airline Route Preference Coordination

Next, the "Intra-Airline Route Preference Coordination" demonstration shows aspects of the intra-airline coordination using the 3D visualization tool. Given FCA constraints, the airline needs to formulate aircraft re-routing preferences. This effort, led by the ATC Coordinator, primarily involves the dispatchers that are handling the "problem aircraft" using the visualization tool and flight planning software to determine aircraft re-route preferences. Dispatchers use layers of information displayed by the airspace visualization tool that help generate the constraint, inputting constraint regions into the flight planning software, and displaying alternate solutions in the airspace visualization tool. Applicable information layers include Navaids, jet routes, forecasted weather, turbulence, predicted congestion, alternate airports, etc. In addition, the ATC Coordinator needs to generate airline routing preferences that maintain bank integrity and minimize the misconnections of passengers, baggage, etc. These bank-related preferences are incorporated into the overall airline routing strategy through a coordination between the ATC Coordinator and airport ramp tower personnel. This coordination involves a telephone exchange between the ATC Coordinator and the Ramp Tower Manager, each referring to their visualization tools.

The Ramp Tower Manager and the ATC Coordinator examine the current gate scheduling software and check the aircraft that need to be re-routed. Figure 14 shows the ATC Coordinator's display showing the current benign destination airport conditions and the gate management software used to identify a behind-schedule aircraft. The Ramp Tower Manager notices that one of the problem aircraft (red field) is significantly behind schedule and checks his Ramp Connect software to determine the magnitude of the possible passenger and baggage misconnections. The Ramp Tower Manager mentions that unless a slot swap occurs, the level of potential misconnections is high. Given the input from the Ramp Tower Manager, the ATC Coordinator intends to ensure that the airline preference needs a slot swap between the behind-schedule problem aircraft and another aircraft. The ATC Coordinator believes that the slot swap should be feasible due to the recent introduction of ATM's CTAS Collaborative Arrival Planning functionality.

ATM-Airline Re-Route Collaboration

Finally, as shown in Figure 15, the "ATM-Airline Re-route Collaboration" demonstration shows aspects of a collaboration between ATM and the airline in arriving at a re-route solution. Having received the recommendation of the slot swap from the Ramp Tower Manager, the ATC Coordinator discusses the problem with the dispatchers over the telephone. They agree to the plan and each dispatcher modifies his individual re-route preferences to speed up the behind-schedule aircraft and slow down and lengthen the flight path for another problem aircraft. If requiring any help in coming up with re-route alternatives, the dispatcher may consult with the ATC Coordinator and they can use the visualization tool to compare alternatives while communicating over the telephone. Each of the dispatchers changes his aircraft re-route preferences and submits a list of prioritized re-route alternatives to the ATC Coordinator through the visualization tool. The ATC Coordinator then collects the prioritized list of re-route alternatives, analyzes the tradeoffs involved with each alternative, and submits the most preferred re-routes to ATM through the visualization tool. The ATM traffic flow managers analyze the re-route preferences and, taking into account other user preferences and forecasted congestion, decide whether the re-routes are acceptable. Eventually, the ATC Coordinator receives ATM approval for all the airline re-route preferences and the dispatchers go back to their roles of flight following the previous "problem" aircraft.

Figure 15 shows the ATC Coordinator's display in the middle of the Airline-ATM re-route collaboration. In addition to the display of weather, FCA, Navaids, and jet routes, a problem aircraft is shown with its nominal flight plan and re-route alternatives. One of the flight plan alternatives was disapproved by ATM due to congestion reasons, while a more circuitous re-route was approved by ATM. ATM approvals/disapprovals are shown using 2D notes. In the case of the ATM disapproval, an audio note is included detailing the reason for the re-route disapproval.

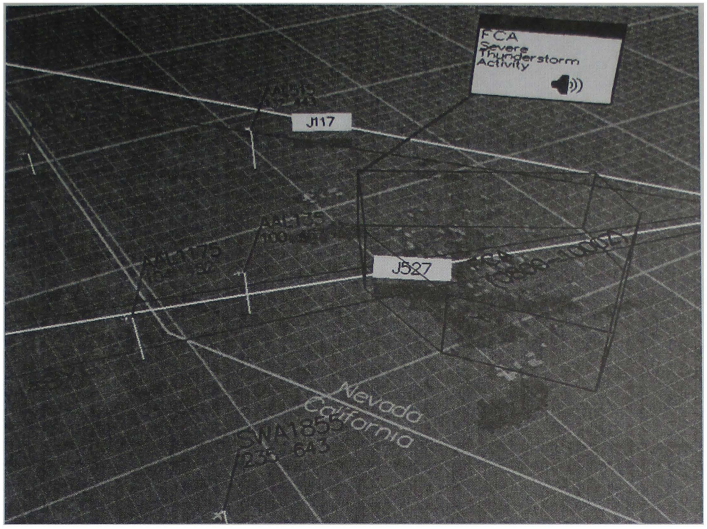


Figure 13. Airspace visualization tool used for ATM-Airline Problem Coordination.

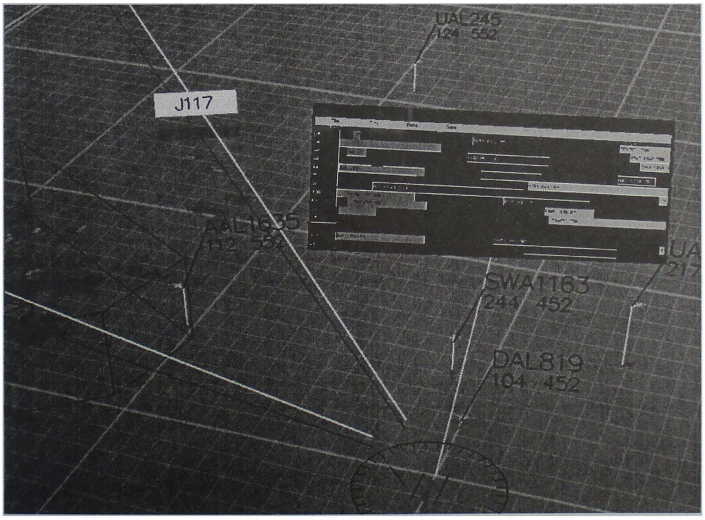


Figure 14. Airspace visualization tool used for Intra-Airline Route Preference Coordination.

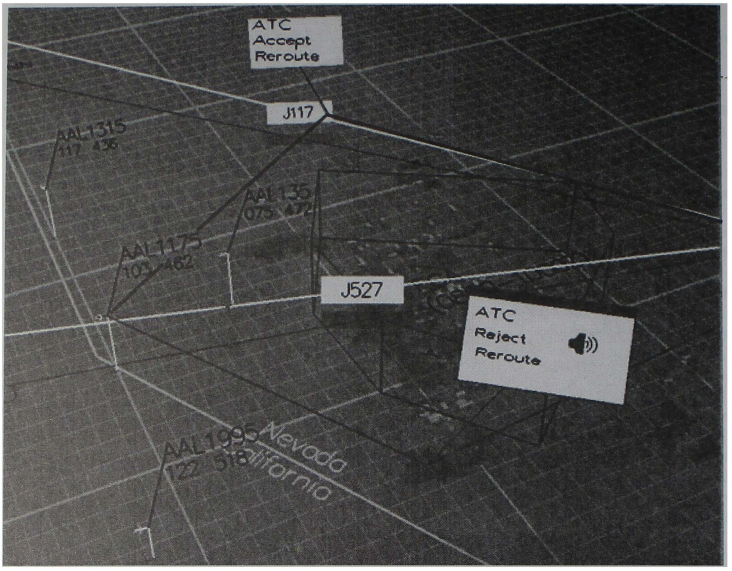


Figure 15. Airspace visualization tool used for an ATM-Airline Re-route Collaboration.

Discussion

From our review of virtual reality collaboration systems, we have identified several systems that demonstrate that virtual reality is feasible both in terms of software (primarily OpenGL graphics) and hardware (Internet and telephone communications) and in terms of bringing two collaborators together for numerous applications. Methods for communication between computers in the collaborative visualization system including, in particular, Sockets and CORBA, have shown that they can fulfill our needs through their cross-platform portability, compliance to industry standards, and previous utility of these technologies for similar applications.

There are many instances where human factors will play a large part in our prototype system. Human factors issues occur at both the local interfacing with icons, symbols, colors, motion, and input/output, as well as the more global functional levels. Human factors experts have recommended to us that these items should be studied in comparison to the human factors literature to identify the most appropriate methods of local interface and to avoid the documented difficulties. The global interface level is concerned with the interaction of two collaborators in terms of the systems ability to facilitate communications, collaboration, and operational objectives. To this end, our collaborative decision making visualization tool should help to create a shared mental model between collaborators[17]. In future work, these human factors issues will be studied as the CDM visualization tool is further developed.

Conclusions

The investigation into a collaborative decision making visualization tool has identified a promising capability that has a need in the airlines industry, will assist in effective and safe transit of aircraft and passengers throughout the NAS, and is feasible to implement with standard Commercial Off The Shelf (COTS) graphics, telephone and Internet communications, and standard symbology and encodings. From the airline perspective, such a visualization system could specifically enable AOC-ATM collaborative routing concepts which would

provide better ATM constraint information and provide a tool to help choose more efficient re-routes. Also, such a tool could help integrate the many decision support systems that are currently in use by airline dispatchers. From the NASA and FAA research perspective, such a visualization system could help enable research into future decision support tools that support collaborative routing, collaborative arrival planning, and collaborative departure scheduling concepts. Having such a tool will allow further research (NASA, FAA, and elsewhere) to progress in evaluating how people can or should collaborate to solve complex problems. These potential uses of our CDM airspace visualization tool, coupled with economic benefits provided by more NAS operations enabled by its use, provide a potentially strong support behind this CDM concept.

Acknowledgments

This research was funded by NASA Ames Research Center under contract NAS2-98059 for the Small Business Innovation Research (SBIR) program. The authors appreciate the program support of Dr. Judith Orasanu at NASA Ames Research Center. The authors also appreciate the suggestions of Dr. Larry Arend from NASA Ames Research Center, Dr. Ron Azuma from HRL Laboratories, and Mr. Mike Daily from HRL Laboratories.

References

[1] Azuma, R., Daily, M., and Krozel, J., "Advanced Human-Computer Interfaces for Air Traffic Management and Simulation," *AIAA Flight Simulation Technologies Conf.*, San Diego, CA, Aug., 1996.

[2] Beatty, R., Corwin, B., and Wambsganss, M., "Collaborative Decision-Making: A Success Story of an Airline-FAA Partnership", *10th Intern. Conf. on Aviation Psychology*, Columbus, OH, May, 1999.

[3] Beyer, R., "Considerations on Graphical User Interfaces for Intelligent ATM Support Systems," *AGARD Meeting on Machine Intelligence in Air Traffic Management*, May, 1993.

[4] Billinghurst, M., Baldis, S., Miller, E, and Weghorst, S., "Shared Space: Collaborative Information Spaces", *HCI International*, 1997.

[5] Billings, C.E., "Toward a Human-Centered Aircraft Automation Philosophy", *International Journal of Aviation Psychology*, Vol. 1, No 4, pp. 261-270, 1991.

[6] Billings, C.E., *Human-Centered Aviation Automation: Principles and Guidelines*, Tech. Memorandum TM-110381, NASA Ames Research Center, Moffett Field, CA, Feb., 1996.

[7] Bowman, D.A. and Hodges, L.F., "Toolsets for the Development of Highly Interactive and Information-Rich Environments", *International Journal of Virtual Reality*, Vol. 3, No. 2, 1998.

[8] Brown, M. A., *On the Evaluation of 3D Display Technologies for Air Traffic Control*, Queen Mary and Westfield College, Dept. of Computer Science, Tech. Report No. 682, Aug., 1994.

[9] Burnett, M.S. and Barfield, W., "Perspective versus plan view air traffic control displays: Survey and empirical results", *Proceedings of Human Factors Society 35th Annual Meeting* , 87-91, 1991.

[10] Carlsson, C. and Hagsand, O., "Dive - A Multi-User Virtual Reality System," *IEEE Virtual Reality Annual Intern. Symposium*, pp. 394-400, 1993.

[11] Daily, M. and Krozel, J., "Human-Centered Interfaces for Air Traffic Control," *40th Annual ATCA Conf. Proceedings*, Las Vegas, NV, pp. 215-219, Sept., 1995.

[12] Dorsky, S., Dow, D., Henthorn, R., and Kondrasuk, G., *Development of a Pseudo Ramp Manager (PRM) Workstation*, Tech. Report 98167-01, Seagull Technology, Los Gatos, CA, July, 1998.

[13] Ellis, S.R., McGreevy, M.W., and R.J. Hitchcock, R.J., "Perspective Traffic Display Format and Airline Pilot Traffic Avoidance", *Human Factors*, Vol. 29, No. 4, pp. 371-382, 1987.

[14] Fuhrmann, A., Loffelmann, H., Schmalstieg, D., and Gervautz, M., "Collaborative Visualization in Augmented Reality", *IEEE Computer Graphics and Applications*, Vol. 18, No. 4, July/Aug., 1998.

[15] Grunwald, A.J., and Ellis, S.R., *Advanced Interactive Display Formats for Terminal Area Traffic Control*, Technical Report, Grant No. NCCW-0045, Technion, Isreal Institute of Technology, Sept., 1995.

[16] Krozel, J., Lee, C., and Mitchell, J.S.B., "Estimating Time of Arrival in Heavy Weather Conditions"" *AIAA Guidance, Navigation, and Control Conf.*, Portland, OR, Aug., 1999.

[17] Krozel, J. and Schleicher, D., "Airspace Visualization Tool for Collaborative Decision Making", *10th Intern. Conf. on Aviation Psychology*, Columbus, OH, May, 1999.

[18] Layton, C., Smith, P., McCoy, E., "Design of a Cooperative Problem-Solving System for En-Route Flight Planning: An Empirical Evaluation," *Human Factors*, Vol. 36, No. 1, pp. 94-119, 1994.

[19] McCann, R.S. and Foyle, D.C., "Scene-Linked Symbology to Improve Situation Awareness", AGARD-CP-575, *Aerospace Medical Panel Symposium*, Brussels, Belgium, April, 1995.

[20] Normand, V. and Tromp, J., "Collaborative Virtual Environments: the COVEN Project," *Proc. of the Framework for Immersive Virtual Environments Conf.*, Pisa, Dec., 1996.

[21] Olvey, D. D., Rhodes, L. R., Carlson, L. S., Reeves, J. J., "Mid Term FAA-Airspace User Collaborative Routing Operational Concept," Contract No. DTAF01-93-C-00001, MITRE Center for Advanced Aviation System Development, McLean, VA, 1998

[22] Orasanu, J., Wich, M., Fischer, U., Jobe, K., McCoy, E., Beatty, R., and Smith, P., "Distributed Problem Solving By Pilots and Dispatchers," in R.S. Jensen (Ed.), *Proc. of the Seventh Intern. Symposium on Aviation Psychology*, Columbus, OH, 1993.

[23] Poupyrev, I., Tomokazu, N., and Weghorst, S., "Virtual Notepad: Handwriting in Immersive VR", *Proc. of the Virtual Reality Annual International Symposium*, pp. 126-132, March, 1998.

[24] Pruyn, P.W. and Greenberg, D.P., "Exploring 3D Computer Graphics in Cockpit Avionics", *IEEE Computer Graphics and Applications* , pp. 28-35, May 1993.

[25] RTCA, Final Report of RTCA Task Force 3, *Free Flight Implementation*, Nov., 1994.

[26] Schleicher, D. R., Couluris, G. J., Sorensen, J. A., *National Airspace System Operational Concept-AATT Products Mapping Analysis*, Appendix C: Collaborative Departure Scheduling, Tech. Report TR98163-01, Seagull Technology, Los Gatos, CA, March, 1998.

[27] Smith, J., Ellis, S., and Lee, E., "Avoidance Maneuvers Selected while Viewing Cockpit Traffic Displays", NASA TM-84269, Oct., 1982.

[28] Smith, P. J., "Brittleness in the Design of Cooperative Problem-Solving Systems: The Effects on User Performance," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 27, No. 3, May 1997.

[29] Smith, P., Billings, C., *et al*, "Can Automation Enable A Cooperative Future ATM System?," *Proc. of the Aviation Psychology Symposium*, 1997.

[30] Smith, P., McCoy, E., Orasanu, J., et al, "Control by Permission: A Case Study of Cooperative Problem Solving in the Interactions of Airline Dispatchers with ATCSCC," *Air Traffic Control Quarterly*, Vol. 4, No. 4, pp. 229-247, 1997.

[31] Tufte, E.R., *The Visual Display of Quantitative Information*, Graphics Press, 1983.

[32] Tufte, E.R., *Envisioning Information*, Graphics Press, 1990.

[33] Tufte, E.R., *Visual Explanations*, Graphics Press, 1997.

[34] Wall, R., Cook, W., DeArmon, J., and Beaton, E., "Self-Managed Arrival Resequencing Tool (SMART): An Experiment in Collaborative Air Traffic Management," *Journal of ATC*, pp. 30-34, April-June, 1997.

[35] Wambsganss, M., "Collaborative Decision Making Through Dynamic Information Transfer," *Air Traffic Control Quarterly*, Vol. 4, No. 2, pp. 109-125, 1996.

AIAA-99-4031

A FLIGHT SIMULATOR STUDY OF ENHANCED/SYNTHETIC VISION FOR HELICOPTER SEARCH AND RESCUE

- Systems Development and En Route Objective Performance

D. E. Sattler* and L. D. Reid†
University of Toronto, Institute for Aerospace Studies
4925 Dufferin St
Toronto, Ontario, Canada
M3H 5T6

This paper describes a preliminary piloted flight simulator investigation into the use of Enhanced/Synthetic Vision System (E/SVS) concepts for the en route phase of a helicopter search and rescue (SAR) mission. Fourteen military helicopter-qualified pilots evaluated an advanced fibre optic helmet mounted display driven by a CAE MaxVue image generator to provide three visual components; an enhanced/synthetic image, aircraft structure normally visible to the pilot and a head-up display presentation. A simulated SAR mission was flown which encompassed piloting aspects that would normally take place between commencement of a search phase (up and away in cruise at 500 feet above ground) and completion of a search phase (near the crash site in a high hover at 50 feet above ground). The independent experimental variables under investigation consisted of the pilot's eyepoint location and the visual database complexity. The experimental measurements included objective performance, pilot questionnaire responses and NASA TLX pilot workload data (the latter two not reported here). The piloting tasks were carefully designed so as to provide guidance in the development of a Canadian flying technology demonstrator to be flown early in the year 2001. Based on the results from this simulator investigation, it would appear that the E/SVS could play an extremely useful role in the helicopter SAR activity. Objective performance data have indicated that an eyepoint location associated with an externally-mounted sensor could provide benefits by reducing minimum obstacle clearance incursions, and that a visual database of reduced complexity could provide a more accurate approach angle to the final hover. Level of pilot experience, technical background and proficiency training were found to influence performance significantly.

Introduction

THE motivation behind development of an enhanced/synthetic vision system (E/SVS) is the desire to provide aircrew with suitable visual cues for flying in conditions of fog, snow, darkness, etc., thereby permitting them to use piloting techniques learned for VMC (visual meteorological conditions). As such, E/SVS would be very useful for helicopter search and rescue (SAR) operations into undeveloped areas during inclement weather. Current thinking holds that E/SVS should be developed to the point where approach minima are 200 ft ceiling and 1/8 statute mile visibility for operations with synthetic vision, and potentially zero-zero for operations with enhanced vision.

In the helicopter SAR application, the E/SVS image is presented to the pilot on a helmet mounted display (HMD) incorporating a head tracker. The "synthetic" part of the image is provided by an onboard computer graphics system containing a visual database of the appropriate area, together with an image generator (IG). The displayed image is registered with the real world using a combination of signals from the head tracker, a precision navigation system and possibly, a helicopter-mounted range finder. The "enhanced" part of the image is produced by helicopter-mounted scanning sensors such as forward looking infrared (FLIR), millimetre-wave radar (MMWR), laser scanners, etc., mounted and controlled so as to follow the pilot's head movement. Obviously, the enhanced image would necessarily present those objects not included in the synthetic image database. The resulting E/SVS display provides the pilot with a fused, composite image incorporating the appropriate features of both the computer-generated and the sensor imagery.

---

* Ph.D. Graduate Student
† Professor and Associate Director

Copyright © 1999 by L.D. Reid. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.



Fig. 1: National Research Council Bell 205 with Prototype EVS

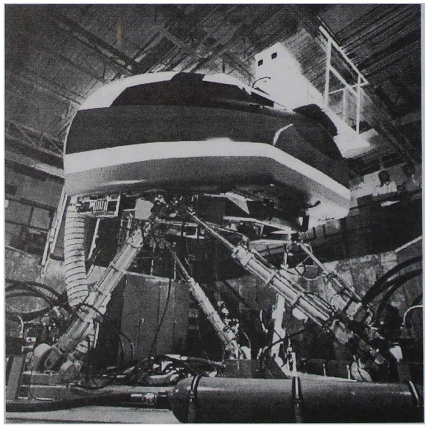


Fig. 2: University of Toronto Institute for Aerospace Studies Flight Research Simulator

The Chief of Research and Development (CRAD) of the Canadian Department of National Defence (DND) has undertaken a research and development program aimed at fielding an E/SVS technology demonstrator to be flown on a Bell 205 helicopter owned and operated by the Canadian National Research Council (CNRC). A SAR mission is to be demonstrated, which involves searching for a downed aircraft in the rugged Gatineau Hills region north of Ottawa, Canada. The CNRC Bell 205 (see Figure 1) has been modified to accept the E/SVS system components, including a head-slewed low-light TV (LLTV) system and a Fibre Optic Helmet Mounted Display (FOHMD) system designed and built by CAE Electronics Ltd of Montreal, Canada. Canadian Marconi Company of Ottawa, Canada is providing digital processing equipment and appropriate systems integration support for the program. A similar FOHMD has been installed in Flight Research Simulator (see Figure 2) at the University of Toronto Institute for Aerospace Studies (UTIAS) in Toronto, Canada. E/SVS experiments carried out at this facility to date[1,2] have provided guidance in support of the eventual flight demonstrations to be conducted in the CNRC Bell 205.

One goal of the second UTIAS experiment[2] as summarized in this paper, was to predict the performance that could be achieved with an E/SVS when applied to the en route helicopter SAR mission, and thus to determine the effects of a number of practical system configuration choices. The underlying E/SVS was assumed to consist of a FOHMD depicting a synthetic image perfectly registered with the outside world and combined in a full field-of-view fashion with an enhanced image that tracked the pilot's head movement in a near-perfect fashion. The resulting fused image appeared genuine to the pilot, with no apparent disparity between the synthetic and the enhanced image elements. The experiment was designed primarily to answer the following helicopter SAR E/SVS questions:

- What performance levels can pilots achieve?
- What impact does visual database detail have on mission performance? (If a simplified database can be employed, it could mean savings in the time required to generate the databases, as well as the cost and complexity of the onboard IG.)
- Does the use of a roof-mounted sensor on the helicopter affect the pilot's ability to carry out the SAR mission? (Because the enhanced image component produced by the onboard sensor will probably have a view-point located other than at the pilot's head position, it will be necessary to generate a fused image on the E/SVS having an eyepoint located at the sensor.)

UTIAS Flight Research Simulator

The UTIAS Flight Research Simulator consists of a cockpit cab mounted on a six degree-of-freedom motion platform[2], as shown in Figure 2. The rear of the cab has been configured as a Bell 205 helicopter cockpit equipped with pedals, a collective lever and centre-stick or side-arm controllers. The simulator is driven by an IBM RISC 6000 Model 390 Power 2 computer, which samples the pilot's control inputs and solves the flight equations using a 60 Hz iteration cycle. An Intel 186/03 Single Board Computer is employed as the motion and sound interface with the RISC. The helmet mounted display system and the magnetic head tracker are interfaced with the RISC via an Ethernet connection. Simulator configuration control, software construction and simulator operation are managed by the CAE Simulation Management Utility (SIMex-PLUS) software[3].

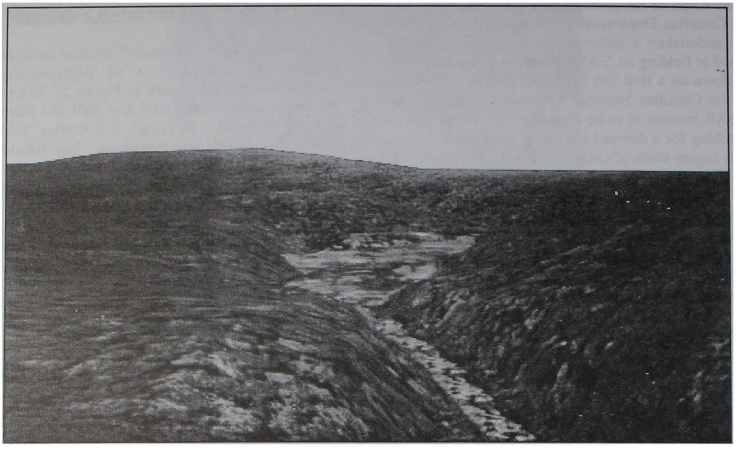
Visuals

For this experiment[2], the pilot had essentially two separate sources of simulated visual information: the FOHMD and the Electronic Flight Instrument System (EFIS). Associated with the former were the E/SVS imagery, the surrounding aircraft structure (hereafter referred to as the cockpit mask, or simply the "mask"), and the head-up display (HUD) flight symbology.

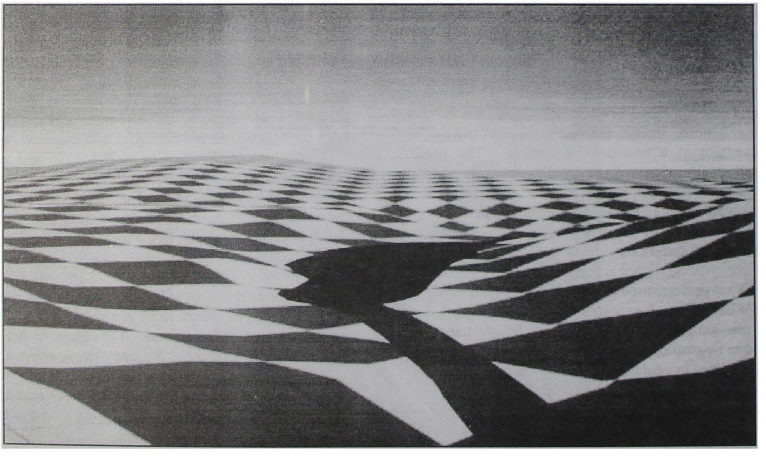
The E/SVS imagery was provided by two visual models representing the same 10 km by 10 km area of land north of Ottawa, Canada known as the "Gatineau Hills" region. These models were developed as separate MaxVue visual databases and executed on the UTIAS MaxVue IG system. The first and most sophisticated model (see Figure 3a), referred to as the "forest" database, incorporated the following features, commensurate with those which would be visually detectable by SAR pilots flying up to five hundred feet above the ground:

- an accurate representation of the terrain elevation at all points inside the defined region;
- appropriate 2D culture models to represent the existence and extent of the many lakes, rivers and roadways;
- 3D culture models representing houses, barns and other objects having a vertical extent too large to represent using 2D culture and
- a forestation texture to represent the densely wooded nature of the region.

The second and least sophisticated model (see Figure 3b), referred to as the "checkerboard" database, was intended to represent a scaled-down yet fully



a: Forest



b: Checkerboard

Fig. 3: Simulated E/SVS Databases

functional version also providing an accurate representation of the terrain elevation, but with a simple checkerboard texture. This latter checkerboard database typically required less than 50% the polygon count of the forest database, with a commensurate reduction in computational requirements.

One of two masks was employed during each simulator run, depending on the selected eyepoint location: an internal "cage" mask (see Figure 4a) for the cockpit eyepoint location, and an external "rooftop" mask (see Figure 4b) for the roof eyepoint location. The HUD flight symbology shown in Figures 4a and 4b was not used in this experiment; instead, an improved version shown in Figure 5 was employed, but only for the roof eyepoint location. Conversely, when the eyepoint was located inside the cockpit, the HUD flight symbology was deselected and only the EFIS arrangement shown in Figure 6 was provided to the pilot (see also Figure 4a).

Fibre Optic Helmet Mounted Display

Installed in the UTIAS Flight Research Simulator is a CAE FOHMD[2,4] (see Figure 7), driven by a 2-channel CAE MaxVue Enhanced B computer IG[5]. The military-style helmet is secured to the pilot's head by a chin strap and positioned as precisely/comfortably as possible with an inflatable liner. Fibre optic cables (often referred to as "ropes") transport the light from the image created by the projectors up to the helmet optical display assembly. Each rope contains 4 million coherent fibres and is approximately 1.8 m (6 ft) in length. The length of the ropes requires the projectors to be located very close to the pilot, in fact, immediately behind the pilot's seat (not shown in Figure 7). In order to reduce the weight of the helmet and display unit supported by the pilot's head, a load relief system is employed. The ropes are supported by fixed lines, while the helmet line is connected via a pulley to a constant tension spring. A six degree-of-freedom magnetic head tracker senses the helmet location/orientation and provides this information to the IG via the head tracker computer. The output from a small lightweight angular rate sensor located atop the helmet is sent to the head tracker computer, where it is used to reduce the system throughput time delay.

Flight Model

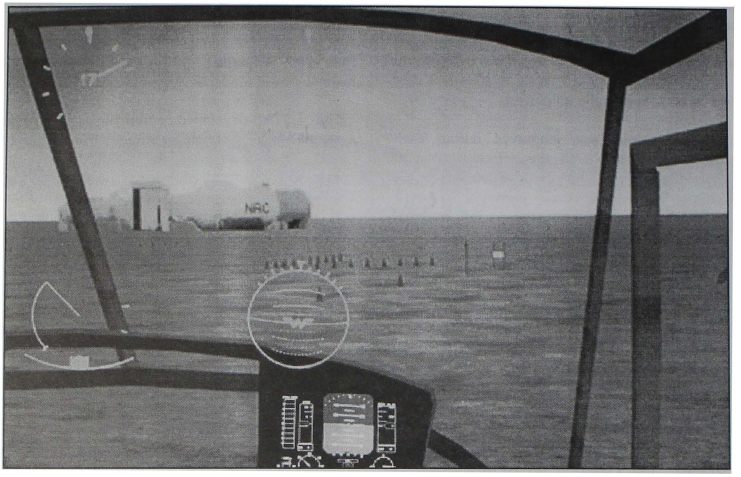
In this experiment, as in several previous UTIAS helicopter simulation experiments[6,7,8,9,10], use was made of a variation of ARMCOP, a generic helicopter simulation computer code originally developed jointly by the NASA and U.S. Army[11]. Following the initial UTIAS endeavor to validate this code[12], ongoing activities have been underway to refine and improve further the code. The current UTIAS-modified ARMCOP model is considered adequate to represent a Bell 205 helicopter for normal operations and manoeuvres of moderate aggressiveness.

The UTIAS-developed automatic flight control system (AFCS) software package, as customized for this E/SVS en route experiment[2], provided an attitude command (AC) response in roll and pitch (see Figure 8a). By definition, an AC system allows the pilot to command roll/pitch attitude in direct proportion to lateral/longitudinal stick displacement. To a large extent, turbulence and aircraft dynamics are automatically suppressed. For this experiment, the yaw channel response was rate command (RC) with turn-coordination and sideslip suppression (see Figure 8b), essentially permitting the pilot to leave his/her feet on the floor while en route. The heave response of the simulated helicopter was not augmented for this experiment.

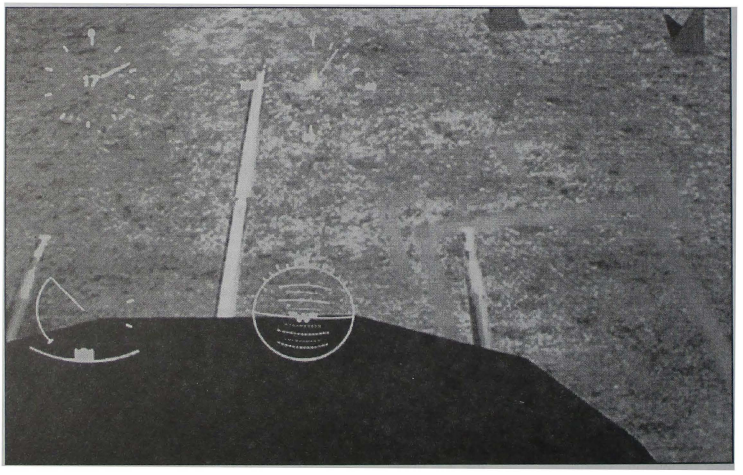
Simulated turbulence was provided by a relatively simple mathematical model[8] which represented turbulence by the sum of nine sinusoidal waves (within the frequency range 0.075 to 2.96 Hz), added to the three body-axes wind components in the simulated flight equations. The spectral composition of all three turbulence components was identical, but the phase angles of the constituent sinusoidal waves were selected randomly. For this experiment, the amplitudes of the resulting turbulence waveforms were scaled to give ±5 kt peak variation in the airspeed components along the x and y body axes, and ±2.5 kt peak variation in the vertical speed component along the z body axis. This turbulence was further superimposed on a steady wind of 10 kt from the southeast.

Motion

One of the original justifications for establishing the UTIAS Flight Research Simulator facility was to undertake the research and development of simulator motion[10,13,14,15,16,17]. With the majority of such activities to-date having specifically examined the fixed-wing application, recent efforts have led to revised algorithms and coefficients which are considered adequate for the current UTIAS-modified ARMCOP model. A dedicated development activity was undertaken to tune the UTIAS Flight Research Simulator motion in preparation for this E/SVS experiment[2], the intent being to maximize simulator motion travel while minimizing simulator motion miscues for the SAR manoeuvres. The basic methodology involved the following:



a: Cockpit



b: Roof

Fig. 4: Simulated E/SVS Masks

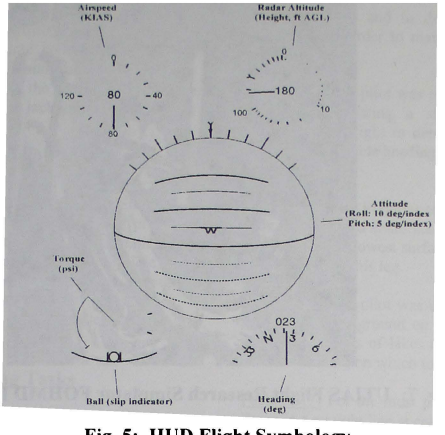


Fig. 5: HUD Flight Symbology

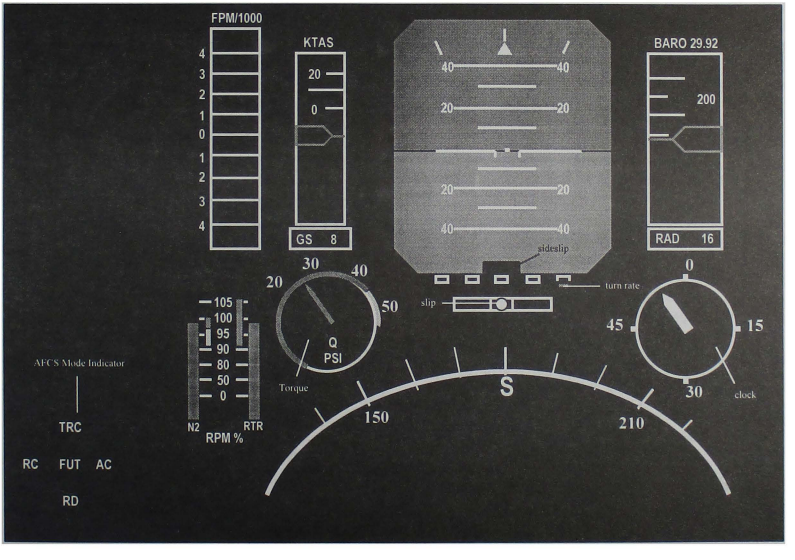


Fig. 6: EFIS Symbology

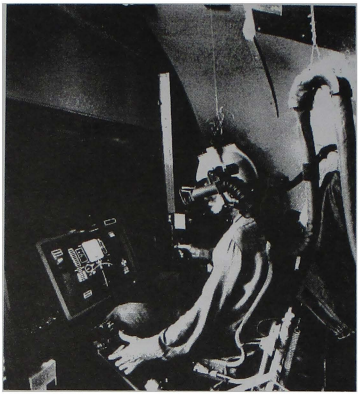
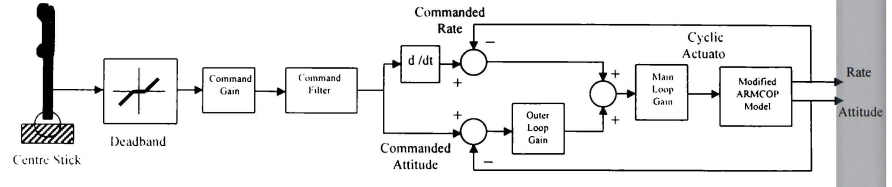
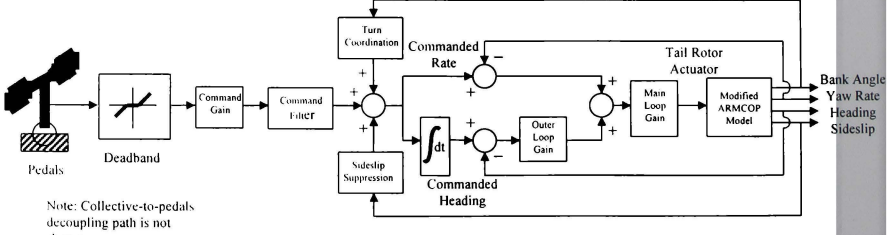


Fig. 7: UTIAS Flight Research Simulator FOHMD Installation



Note: "Outer-outer control loops and transformations required to re-reference between inertial and helicopter axes are not shown.

a: Roll/Pitch Attitude Command



Note: Collective-to-pedals decoupling path is not shown.

b: Yaw Rate Command with Turn Coordination and Sideslip Suppression

Fig. 8: Simulated Automatic Flight Control System

- pre-recording representative manoeuvres,
- establishing individually the "maximum open" values for gain and time constant coefficients in each motion degree of freedom and
- collectively consolidating the coefficients gains (i.e., closing-down") subject to jack limit considerations.

Sound

In all UTIAS flight simulator experiments, cockpit sound is simulated by an E-mu System Inc Emax digital sampling keyboard[6]. For this experiment, previously recorded and digitized actual Bell 205 cockpit sounds representing different flight conditions (e.g. idle, hover and cruise) were replayed in real-time during the simulation. Although the system is capable of stereophonic sound, in its present implementation only monophonic sound is used.

Piloting Tasks

The piloting tasks employed in this flight simulator experiment[2] were developed for the sole purpose of exercising the simulated E/SVS system under study. Every effort was made, however, to keep the tasks representative of those which might be employed in an en route helicopter search phase for a downed light float plane. A nominal ground track was established encompassing a cruise task, a recce task and an approach task, and each evaluation pilot was ultimately trained to follow this track. Refer to the map contained in Figure 9, which depicts the en route course and numbered locations of specific interest.

Cruise

The objective of the cruise task was to "contour fly" at 300 ft above the lowest terrain as close as possible along the nominal track, while maintaining a 90 kt airspeed.

Position (1) The cruise task started with the helicopter approximately trimmed for flight at 300 ft above ground level (AGL) and 90 kt indicated airspeed (KIAS), approaching Mullin Marsh on the desired westerly heading.

Position (2) As the helicopter overflew Mullen Marsh, the pilot was required to make a heading correction of approximately 20° right and initiate a slight climb due to rising terrain.

Position (3) The river bed diverged to the right of desired track, and the aircraft overflew the "first saddle". The pilot was required to increase collective on the upslope, and to decrease collective on the downslope, in order to maintain the target height of 300 ft AGL.

Position (4) The pilot was required to perform a right turn while initiating a slight descent and while scanning ahead/right to determine the lowest ground and hence a suitable heading on which to roll-out from the turn.

Position (5) The irregular terrain resulting from river beds, swamps and small lakes made selection of track (i.e., the path of lowest surface elevation) a demanding piloting task on this leg.

Position (6) The pilot was required to perform a turn around the high ground on the right side, to visually acquire the series of lakes and thence to determine a suitable heading on which to roll-out from the turn.

Position (7) For the most part, this segment could be flown in a straight line at constant altitude.

Position (8) The pilot was required to perform a right turn so as to overfly three, progressively smaller lakes, then immediately perform a tight left turn to align the aircraft with the required distant river bed.

Position (9) The river bed was initially absent, leading up to the "second saddle". The pilot was required to increase collective on the upslope, and to decrease collective on the downslope. An added complication was the difficulty associated with establishing the required track (based on an off-boresight, distant river bed) for the next leg.

Position (10) The original deep river bed oriented left-to-right was overflown, and the pilot was required to decrease collective on the downslope and to increase collective on the upslope. Once over the downslope, the distant river bed defining the required track was more discernable. Once over the upslope, a protracted rate of climb was required to accommodate continuously rising ground.

Position (11) The pilot was required to perform a left turn so as to overfly the centres of several small marshes connected by the river bed.

Position (12) (Hidden within the municipal boundary labelling on the map.) This position is employed herein to demonstrate the procedure associated with the pilot providing estimates of range, height difference and relative bearing of a given visual feature. In this particular case, the feature under consideration is the peak of the hill on the inside of the

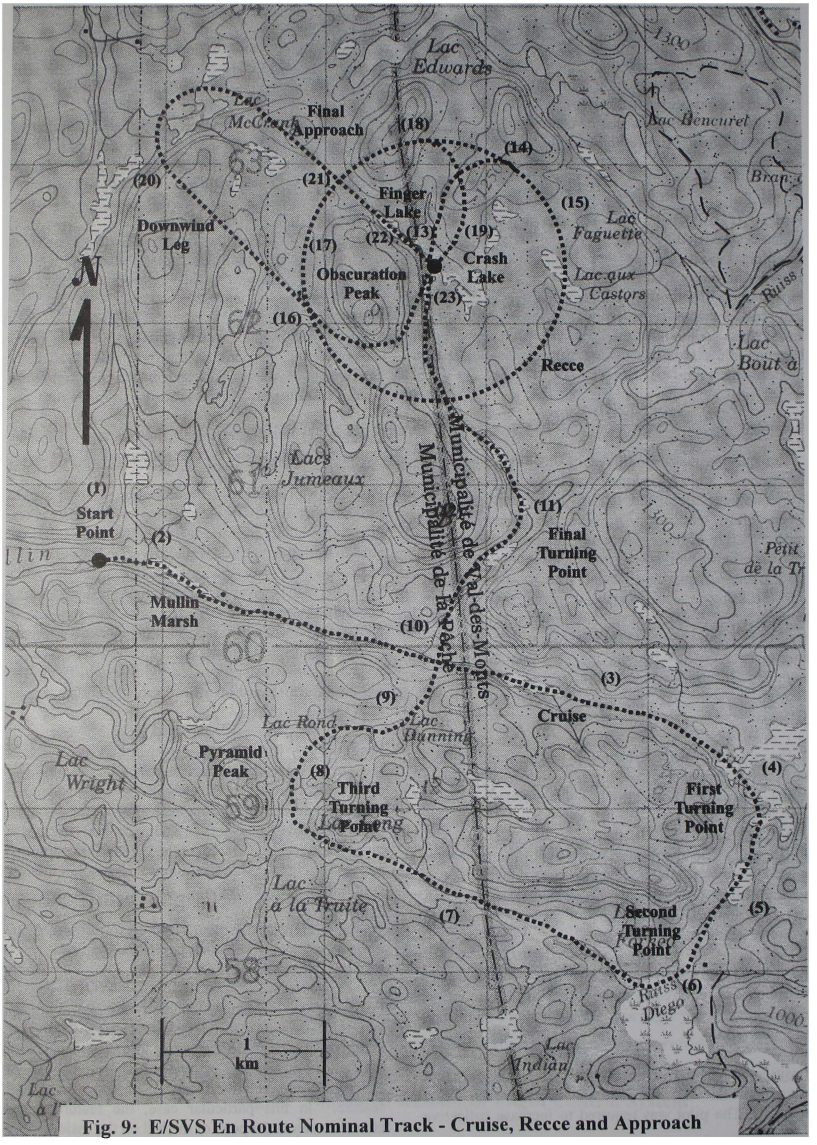


Fig. 9: E/SVS En Route Nominal Track - Cruise, Recce and Approach

final turning point (11). As the turn was commenced, the experimenter asked the pilot for numerical values as follows:

- range to the peak
- height of the peak above/below eye-level
- clock position of the peak

Position (13) Upon over flying Crash Lake, the data recording process was terminated and the pilot was advised to transition to the recce pattern.

Recce

Position (14) The pilot was required to perform a transition from the cruise task to the recce task. This involved increasing height from 300 ft AGL to 500 ft above the lake surface, decreasing airspeed from 90 KIAS to 60 KIAS and establishing the aircraft in circular pattern of radius 1/2 nm about the crash site. This transition was not a formal part of the experiment, i.e., quantitative data were not collected during the transition.

Position (15) The transition was considered complete upon reaching the required flight parameters of 60 KIAS, 500 ft above Crash Lake and 1/2 nm radius from the crash site.

Position (16) Upon reaching the southwest quadrant, the pilot would typically lose visual reference to Crash Lake, due to the presence of Obscuration Peak. (Climbing was not an option, due to the presumed cloud base.)

Position (17) The pilot re-acquired visual references to Crash Lake, initiated corrective action and continued with the recce.

Position (18) Upon closing the circular recce pattern, the data recording process was terminated.

Position (19) The pilot was required to perform a low-and-over of the crash site area (i.e., height less than 100 ft AGL and airspeed greater than 45 KIAS). This position is employed herein to demonstrate the procedure associated with the pilot providing a size-estimate of a given visual feature. As the manoeuvre progressed, the experimenter asked the pilot for the required numerical value (for example, the width of Crash Lake abeam the crash site). Other than the size estimates, no other quantitative data were collected during the low-and-over.

Approach

Position (20) The pilot was required to perform a transition from the low-and-over to an initial approach condition of 60 KIAS, 300 ft above Crash Lake, 1/2 nm from the crash site, with Finger Lake and Pine Point approximately aligned. (Pine Point is not shown on the map, but was in close proximity to the crash debris) This transition was not a formal part of the experiment, i.e., quantitative data were not collected during the transition.

Position (21) The pilot was required to establish the helicopter at the 1/2 nm approach fix with an airspeed of 60 KIAS, a height of 300 ft above Crash Lake and the tip of Finger Lake approximately aligned with the tip of Pine Point.

Position (22) The pilot was required to perform an approach of constant azimuth to the southeast and constant vertical flight path of approximately four and one-half degrees while maintaining a "constant sight picture" and an "apparent brisk walking pace".

Position (23) Upon reaching approximately 600 ft short of Pine Point (100 ft AGL), the data recording process was terminated. The pilot was required to transition the helicopter to a high hover at 50 ft AGL, with Pine Point (more specifically, a large pine tree located on the point) at a clock position of 1:30 and a distance of 50 ft. This transition was not a formal part of the experiment, i.e., quantitative data were not collected during this final transition to hover.

Piloting Activities

Each pilot progressed through the following experimental process in the sequence indicated below[2]:

- briefing,
- familiarizing,
- demonstrating,
- training and evaluating.

The briefing consisted primarily of reviewing a document[18] supplied in advance to each of the participating pilots. Contents included information on the simulated aircraft characteristics, the experimental procedure and the simulator safety procedures.

Familiarizing runs were intended to introduce the pilot to the simulator and the accompanying visual systems. During the first of two 20-minute familiarizing runs, both turbulence and simulator

motion were absent, the full EFIS display was present, the pilot's eyepoint was located in the cockpit and the forest visual database was used. During the second familiarizing run, turbulence and simulator motion were introduced.

The above familiarization activity was followed by two 15-minute demonstrating runs. These runs were essentially replays of previously recorded UTIAS runs, and were designed to demonstrate important aspects of tasks and experimental configurations that the pilots would ultimately be evaluating. During these demonstrating runs, the pilot occupied the simulator cockpit in a normal fashion, observing the activity, much as a co-pilot or passenger would.

Finally, during the training/evaluating runs, each pilot was required first to "train-up" with the configuration to be evaluated. He/she would initially perform one or two training runs followed by one evaluation run, the latter being used to generate the actual performance data used for analysis. During these runs, altitude information (i.e., pressure altimeter, radar altimeter and vertical speed) were purposely absent or presented momentarily only, except in the case of the first training run when two such training runs were undertaken.

Measurements

Both objective and subjective measurements[2] were collected during/after the evaluation runs. Only the former are reported in this paper, although the latter are briefly discussed below for purposes of completeness.

Objective

The following eight variables were selected for use as objective measurements, each involving values sampled over the specified portions of a run:

- cruise track error,
- cruise height above ground,
- cruise airspeed,
- recce radius,
- recce height above lake,
- recce airspeed,
- approach vertical angle and
- approach horizontal angle.

In each case, the mean value, standard deviation, minimum and maximum values (designated as *Mean*, *S.D.*, *Maximum* and *Minimum*) of the performance variable were computed on-line, by the simulator host computer. As it turned out, performance for cruise airspeed and recce airspeed were non-issues, and are not discussed further in this report.

Subjective

After each session in the simulator, the pilot was required to fill out a *Post-Session Questionnaire*[2,18]. An additional (and different) *Post-Segment Questionnaire*[2,18] was required to be filled out following completion of the entire en route activity. Questions were asked about the helmet and associated hardware, the FOHMD optics/imagery and the extent to which the simulated E/SVS activity compared to real flight. In addition to providing a numerical rating for each question, pilots were encouraged to express their comments, both written and oral (the debriefings were tape-recorded). A separate *Psycho-Physiological Response Questionnaire*[2,18] was administered only if the pilot experienced discomfort/disorientation. The pilots were also instructed to indicate their workload rating for the task by placing a mark on the scale at the appropriate location on the *NASA TLX Workload Contributor Rating Sheet*[2,18,19]. Upon completion of the entire en route activity, each pilot further ranked the sources of workload using the *NASA TLX Workload Contributor Comparison Sheet*. The preceding workload rating and comparison data were subsequently used by the experimenter to arrive at the final NASA TLX numerical ratings. As previously mentioned, results from subjective measurements are not presented in this paper.

Results and Discussion

Due to the relatively small sample size of subject pilots (i.e., 14 evaluation pilots) and the intricate nature of the piloting tasks, traditional methods were largely employed to analyze the objective performance data for the en route task[20]. (ANOVA was employed for an initial analysis[2], however the technique proved too superficial for the detail required.)

Figure 10 illustrates the upper limits for defining satisfactory objective performance, as adopted for this experimental analysis. Percentages for μ (i.e., average values across the 14 pilots) and σ (i.e., standard deviation among the 14 pilots) were utilized for performance assessment wherever possible, in accordance with accepted military operating and training criteria, respectively. For example, based on a target value of 100, the satisfactory μ upper limits across the evaluation pilot population would be:

- $Mean_{\mu}$; 110, or 10 % over the target value
- $S.D._{\mu}$; 121, or an additional 10 % of $Mean_{\mu}$
- Max_{μ}; 132, or 200 % of $S.D._{\mu}$ over $Mean_{\mu}$

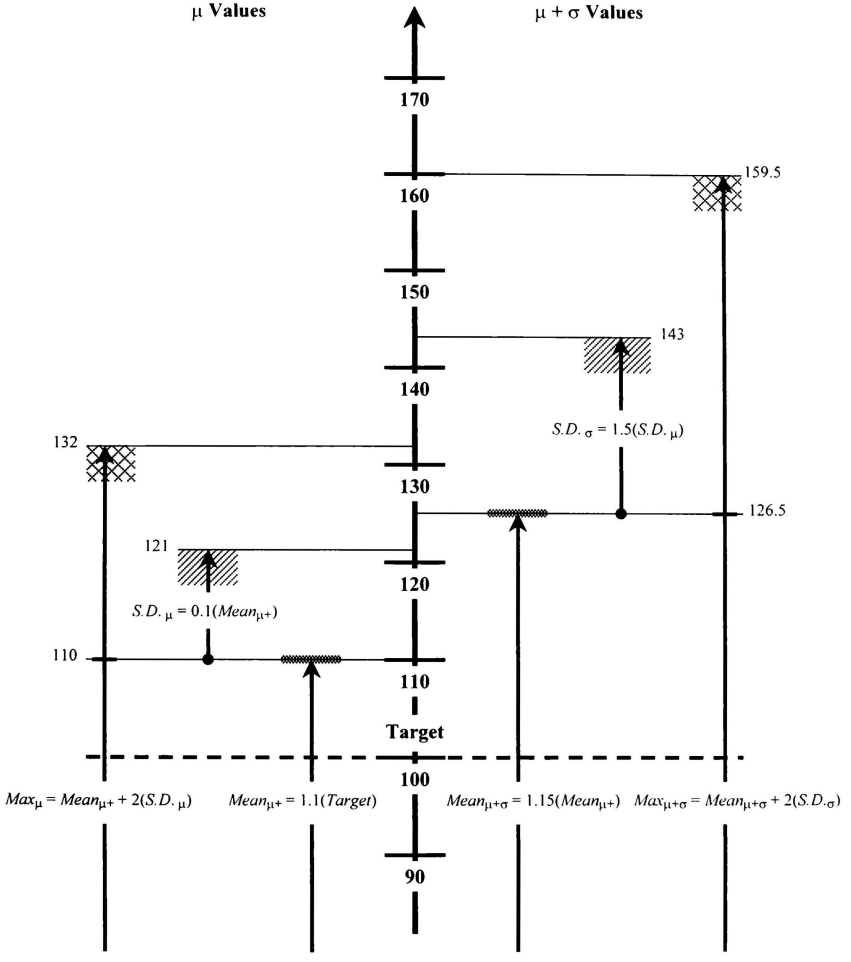


Fig. 10: Limits for Satisfactory Performance

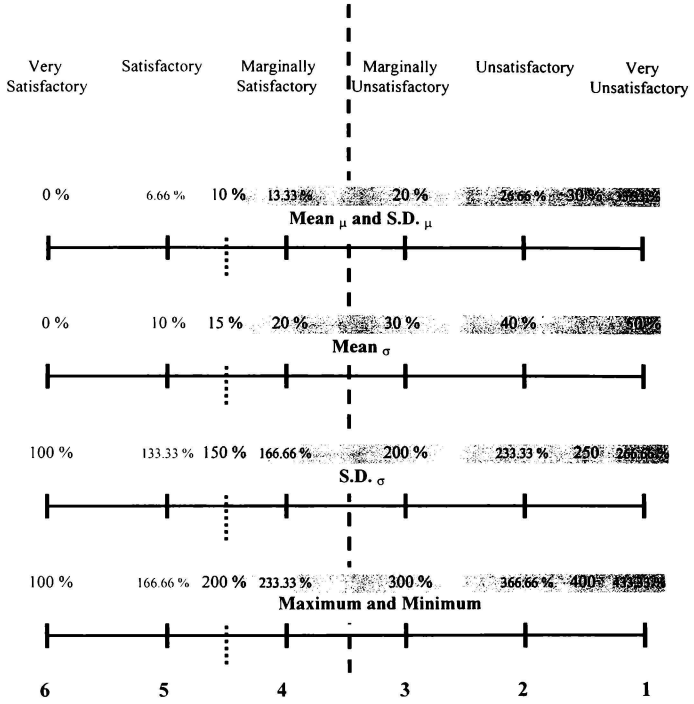


Fig. 11: Level/Rating Scales for Performance Values

Satisfactory σ limits among the evaluation pilot population would be:

- $Mean_{\mu+\sigma}$; 126.5, or 15 % over $Mean_{\mu}$
- $S.D._{\sigma}$; 143, or an additional 150 % of $S.D._{\mu}$
- $Max_{\mu+\sigma}$; 132, or 200 % of $S.D._{\sigma}$ over $Mean_{\mu+\sigma}$

Lower limits for defining satisfactory objective performance were based on the same percentages[20] (i.e., 10 %, 15 %, 150 % and 200 %). For *Cruise Track Error* and *Approach Horizontal Angle*, where percentages were not always applicable, extensive use was made of UTIAS staff (both engineers and pilots) to assess the levels of performance on a one-by-one basis.

Other levels of performance (i.e., below satisfactory), as well as the numerical rating scales associated with the performance levels are given in Figure 11. Essentially, the additional levels/ratings therein are based on proportionality, with six divisions covering the entire range from very satisfactory (Rating 6) to very unsatisfactory (Rating 1).

Table 1 presents a summary of the en route objective performance values obtained from this experiment. The following two sets of performance values were calculated for each variable/parameter[20]:

- achieved performance, calculated by taking the average over all configurations/measures and
- projected performance, calculated by taking the average over specific configurations/measures, so as to provide improved performance.

Grey-shaded values in Table 1 represent less than satisfactory performance. For the unsatisfactory achieved performance values, difference data was utilized to determine which configurations/measures should be rejected/retained in order to improve the performance, thereby giving the projected performance values.

Difference data for this experiment were calculated for a total of six measures, as follows:

- $\Delta\mu$ between configurations, for which 15 % is considered significant (see Figure 12),
- $\Delta\sigma$ between configurations, for which 15 % is also considered significant (see again Figure 12),
- skew between σ+ and σ- (i.e., asymmetric σ values), for which 15 % is also considered significant,
- normality of the distribution (i.e., variation from a Gaussian distribution), for which 1 data point wide is considered significant (see Table 2),
- dependence on Group 1 pilots (those with less flying experience and/or technical background, abbreviated as G1), for which 1 data point is considered significant (see again Table 2) and
- dependence on Run 1 evaluations (those runs where the pilot was evaluating for his/her first time, i.e., with minimum exposure to the UTIAS simulator and the simulated task, abbreviated as R1), for which 1 data point is considered significant (see again Table 2).

Table 3 presents a summary of the en route objective performance differences obtained from this experiment. Grey-shaded values in Table 3 represent significant differences that supposedly could be utilized to improve the performance in Table 1.

Returning now to Table 1, note that three variables exhibited less than satisfactory achieved performance levels: *Cruise Track Error*, *Cruise Height Above Ground* and *Approach Vertical Angle*. Each of these cases is discussed separately below.

For *Maximum* and *Minimum Cruise Track Error*, performance values for both μ (i.e., across the pilots) and σ (i.e., among the pilots) for were found to be marginally unsatisfactory[20]. Figure 13 illustrates the best and worst case runs for this variable, from which the performance problem was obviously caused by "overshooting" the turn (i.e., pilot control action was not initiated until well after that instant in time which would have resulted in proper following of the reference track). From Table 3, the data were dependent on both G1 and R1 measures, hence, eliminating pilots with lesser experience/background and providing sufficient training could be expected to improve the performance. This has been verified in Table 1 for the projected performance values, which are satisfactory, although marginally so.

Although performance for the *Minimum Cruise Height Above Ground* was within the limits imposed by Figures 10 and 11 (i.e., 200 %), closer scrutiny[20] revealed that in five of the fifty-six evaluation runs, heights above ground deviated momentarily below 150 ft. In consideration of the fact that these potential "obstruction clearance incursions" occurred mostly over high ground (where additional man-made obstacles are often located, e.g., microwave towers, which are typically 150 ft high), the performance was deemed to be unsatisfactory, for reasons of flight safety. It is worthwhile noting that in each of the five aforementioned cases, G1 and/or R1 data were involved, and that pilots on the whole tended to fly at slightly higher heights with the roof eyepoint location.

Table 1: Summary of En Route Objective Performance Values

Variable/ Parameter	Achieved Performance Values				Projected Performance Values		
	Cause	Worse	Satisfaction	Rating	Improvement	Satisfaction	Rating
Cruise Track Error							
- *Mean*			very satisf	6	no G1, no R1	very satisf	6
- *S.D.*	μ		marg satisf	4 *		marg satisf	4 *
- *Maximum*	μ σ	Roof	marg unsatisf	3 * *		marg satisf	4 *
- *Minimum*	μ σ	Cockpit	marg unsatisf	3 * *		marg satisf	4 *
Cruise Height Above Ground							
- *Mean*			satisf	5	no G1, no R1 Roof	satisf	5
- *S.D.*	μ σ	Cockpit	marg satisf	4 *		marg satisf	4 *
- *Maximum*	μ σ	Cockpit	marg satisf	4 *		marg satisf	4 *
- *Minimum*	μ	Cockpit	unsatisf	2 * * *		marg satisf	4 *
Recce Radius							
- *Mean*			satisf	5		satisf	5
- *S.D.*	μ		marg satisf	4 *		marg satisf	4 *
- *Maximum*			satisf	5		satisf	5
- *Minimum*			satisf	5		satisf	5
Recce Height Above Lake							
- *Mean*			satisf	5		satisf	5
- *S.D.*	μ	Roof	marg satisf	4 *		marg satisf	4 *
- *Maximum*			satisf	5		satisf	5
- *Minimum*			satisf	5		satisf	5
Approach Vertical Angle							
- *Mean*	μ σ	Forest Checkerboard	very unsatisf marg unsatisf	1 * * * * 3 * *	Checkerboard	marg unsatisf	3 * *
- *S.D.*	μ σ	Checkerboard Forest	unsatisf marg satisf	2 * * * 4 *		unsatisf	2 * * *
Approach Horizontal Angle							
- *Mean*			satisf	5		satisf	5
- *S.D.*	σ	Cockpit	marg satisf	4 *		marg satisf	4 *

marginally satisfactory * marginally unsatisfactory * * unsatisfactory * * * very unsatisfactory * * * *

data suspect due to start and end effects

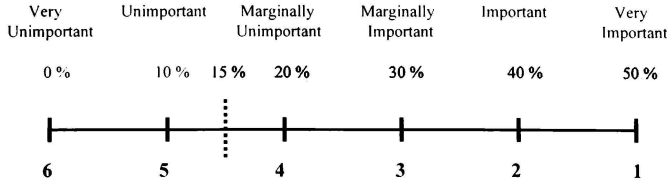


Fig. 12: Level/Rating Scales for μ and σ Performance Differences

Table 2: Distribution Percentages for 14/4 Experiment*

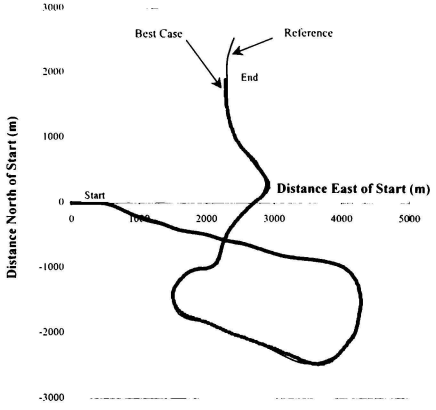
Distribution	% of Total Data Points Outside	
	Single σ-/+ Boundary	Double σ-/+ Boundaries
Normality		
- 2 Data Point Narrow	8.8 %	18 %
- 1 Data Point Narrow	12 %	25 %
- **Normal**	**16 %**	**32 %**
- 1 Data Point Wide	19 %	39 %
- 2 Data Point Wide	23 %	46 %
Group 1 (G1) Dependency		
- **Independent**	**4.5 %**	**9.1 %**
- 1 Data Point Dependent	8.1 %	16 %
- 2 Data Point Dependent	12 %	23 %
Run 1 (R1) Dependency		
- **Independent**	**4 %**	**8 %**
- 1 Data Point Dependent	7.5 %	15 %
- 2 Data Point Dependent	11 %	22 %

* Experimental conditions: 14 evaluators, 4 independent variables

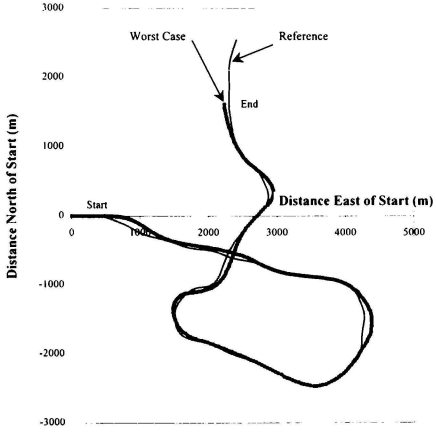
Table 3: Summary of En Route Objective Performance Differences

Variable/ Parameter	Observed Performance Differences								
	Measure						Worse	Importance	Rating
Cruise Track Error									
- *Mean*				R1				unimport	5
- *S.D.*	skew+		G1			σ	Cockpit	marg unimport	4 *
- *Maximum*			G1	R1		σ	Roof	marg unimport	4 *
- *Minimum*	skew-			R1		σ	Cockpit	important	2 * * *
Cruise Height Above Ground									
- *Mean*	skew+							very unimport	6
- *S.D.*	skew+					σ	Cockpit	marg unimport	4 *
- *Maximum*	skew+					σ	Cockpit	marg unimport	4 *
- *Minimum*			G1			σ	Roof	marg unimport	4 *
Recce Radius									
- *Mean*		wide				σ	Cockpit	marg unimport	4 *
- *S.D.*	skew+					σ	Forest	marg import	3 * *
- *Maximum*	skew+			R1		σ σ	Roof Forest	marg unimport marg import	4 * 3 * *
- *Minimum*						σ	Forest	marg unimport	4 *
Recce Height Above Lake									
- *Mean*	skew+					σ	Cockpit	marg unimport	4 *
- *S.D.*	skew+		G1			μ σ σ	Roof Roof Checkerboard	marg unimport import marg unimport	4 * 2 * * * 4 *
- *Maximum*	skew+		G1	R1		σ σ	Cockpit Checkerboard	marg unimport marg unimport	4 *
- *Minimum*						σ	Checkerboard	marg unimport	4 *
Approach Vertical Angle									
- *Mean*	skew-	wide				μ	Forest	marg import	3 * *
- *S.D.*	skew+					μ	Checkerboard	import	2 * * *
Approach Horizontal Angle									
- *Mean*	skew+			R1		σ	Cockpit	marg import	3 * *
- *S.D.*	skew+					μ σ	Cockpit Cockpit	marg unimport import	4 * 2 * * *

marginally unimportant * marginally important * * important * * * very important * * * *



a: Best Case



b: Worst Case

Fig. 13: Best and Worst Case Cruise Track Error

Hence, eliminating pilots with lesser amounts of experience/background/training, and retaining the roof eyepoint location only could be expected to improve the performance, as verified in Table 1 where projected performance values are marginally satisfactory. Figure 14 illustrates best and worst case runs, with the latter illustrating excessive ground clearance (as opposed to the inadequate ground clearance discussed above).

Finally, the *Approach Vertical Angle* provided the least satisfactory performance of all the variables examined, with μ performance levels down to very unsatisfactory for *Mean* and unsatisfactory for *S.D.* As annotated in Table 1 however, the data was suspect due to uncertainties associated with start and end point effects. Figure 15 illustrates two sample cases of the *Approach Vertical Angle* variable. In the first case, the pilot over-estimated the half-mile distance, which consequently resulted in a steep approach; whereas in the second case, the pilot under-estimated the half-mile distance, which resulted a shallow approach. Hence, the vertical approach angle was often more a function of feature recognition (used frequently in lieu of distance estimation) at approach initiation, rather than sight picture during the approach, per se. Also, the physics of approach angle measurement result in increasing sensitivity with decreasing distance to the final end point. The consequence of this was a mostly-monotonic diverging angle near the end of the approach, the sense of which depended on whether the approach angle was initially on the steep or the shallow side. Finally, the fact that the helicopter was continuously decelerating during the approach caused data near the end to be more heavily weighted in the statistical calculations than data near the start. Because of these start/end effects, the vertical approach performance data for this experiment were considerably suspect as to their validity. However, should the data incidentally prove valid, indications are that excessively steep *Mean Approach Vertical Angle* achieved performance values would occur across the pilot population when using the forest database[20]. From Table 3, this problem could be considerably alleviated by utilizing the checkerboard database only, however, the projected *S.D. Approach Vertical Angle* performance would then be, at best, less than satisfactory. These projected performance values are recorded in Table 1.

Conclusions

The concept of enhanced/synthetic vision as applied to the helicopter search and rescue mission was successfully demonstrated and examined in a preliminary flight simulation experiment[2], the enroute portion of which has subsequently been analyzed[20] and is reported herein. Given certain system configurations and pilot qualifications, objective performance was found to be mostly satisfactory, although marginally so, with the exception of one variable, which was found to exhibit potentially unsatisfactory performance.

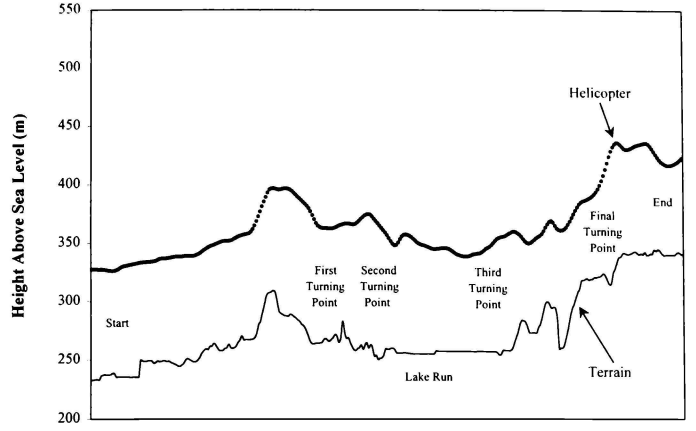
Satisfactory objective performance for horizontal tracking during cruise was found to require a skill level commensurate with those pilots having higher levels of flying experience, technical background and proficiency training. Otherwise navigation turning points would frequently be "overshot", causing excessively large maximum and minimum excursions. No overall benefit could be derived from utilizing the specific eyepoint locations or database types investigated in this experiment.

For the vertical task during cruise, incursion of the minimum obstacle clearance height gave rise to flight safety concerns in approximately ten percent of the evaluation runs. However, satisfactory objective performance could be guaranteed by utilizing pilots with higher skill levels, in conjunction with an eyepoint location on the roof. The latter requirement arose due to the fact that pilots, on the whole, tended to fly at slightly higher heights with the roof eyepoint location than with the cockpit eyepoint location.

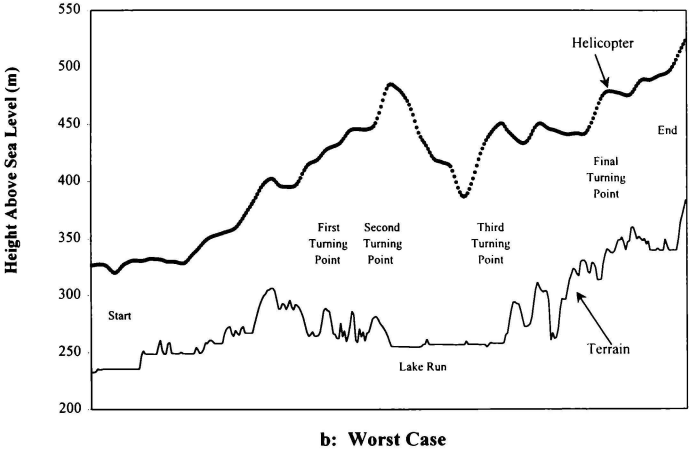
For the recce task, satisfactory objective performance was somewhat compromised by deviations about the target values for radial distance and height. While although not excessive in magnitude, these deviations were quite long in duration, commensurate with the pilots tending to "wander" within otherwise acceptable maximum/minimum boundaries. No tangible benefits could be derived from employing specific E/SVS configurations and/or pilot qualifications.

Data for the approach task are currently suspect, due to uncertainties associated with start and end point effects. It is recommended that additional simulation experiments be conducted using improved data collection procedures and analysis techniques to re-examine primarily the vertical approach task. Preliminary indications are, however, that excessively steep approaches can be expected across the pilot population when using the forest database. If the checkerboard only were used, mean approach angles would likely be satisfactory, but angle deviations could become excessively large.

In closing, it is worthwhile noting that everyone involved in this enhanced/synthetic vision simulation experiment, particularly the fourteen military

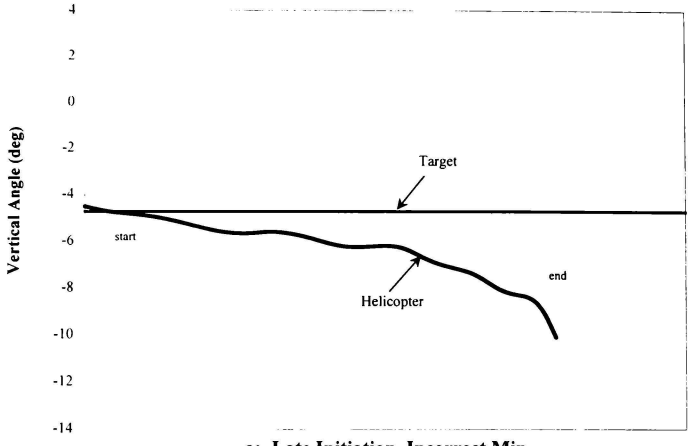


a: Best Case

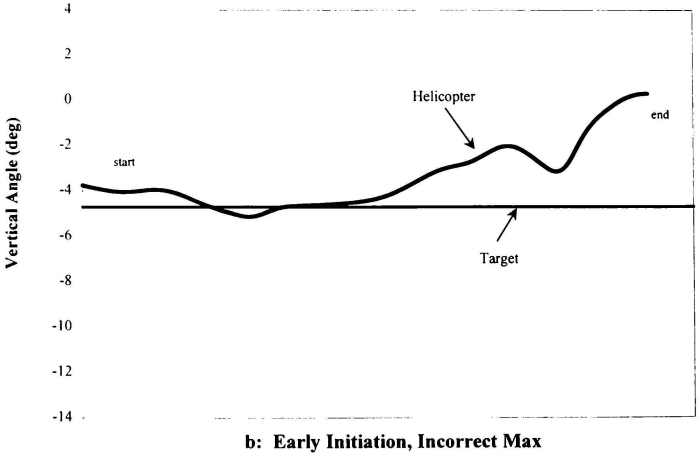


b: Worst Case

Fig. 14: Best and Worst Case Cruise Height Above Ground



a: Late Initiation, Incorrect Min



b: Early Initiation, Incorrect Max

Fig. 15: Samples Approach Vertical Angle

evaluation pilots, felt that the enhanced/synthetic evaluation pilots, felt that the E/SVS concept has tremendous potential for expanding and improving the helicopter role for search and rescue applications. Consequently, further flight simulation experiments have taken place recently and are planned to continue at the University of Toronto Institute for Aerospace Studies.

Acknowledgements

The research reported herein was made possible through financial support from the Canadian Department of National Defence, under contract designation W8477-6AC22/01SV. The authors wish to express their sincere appreciation for technical and other support provided by participating organizations, including the Centre for Research and Space Technology, the Canadian National Research Council, CAE Electronics Ltd and Canadian Marconi Company. The contribution of the fourteen Canadian military helicopter pilots who served as evaluation subjects is gratefully acknowledged, as is the contribution of all the participating support staff and graduate students from the University of Toronto Institute for Aerospace Studies, Flight Simulation Laboratory.

References

[1]Tai, E., *A Preliminary Evaluation of Synthetic Vision for Helicopter Search and Rescue*, University of Toronto Institute for Aerospace Studies, M.A.Sc. Thesis, 1998.

[2]Reid, L.D., Sattler, D.E., Graf, W.O., Dufort, P.A., and Zielinski, A.W., *Cockpit Technology Simulation Development Study for Enhanced/Synthetic Vision System*, University of Toronto Institute for Aerospace Studies, Report No. 355, November 1998.

[3]CAE Electronics Ltd, *SIMex-PLUS Users Guide*, Volume 4, CAELIB on UNIX Systems, CAE Electronics Ltd, Saint-Laurent, Quebec, 1994.

[4]Reid, L.D., Graf, W.O., *Human/Machine Interaction Aspects of Helmet Mounted Display*, University of Toronto Institute for Aerospace Studies, Final Report to Department of National Defence Contract W2207-2-AF24/01-SV, March 1995.

[5]CAE Electronics Ltd., *Image Generator for the UTIAS Flight Research Simulator: Technical Specification*, TDP 09523 Revision 1, CAE Electronics Ltd, Saint-Laurent, Quebec, October 1994.

[6]Reid, L.D., Advani, S., and de Leeuw, J.H., *An Evaluation of Decelerating IFR Approaches Utilizing a Helicopter Flight Simulator*, University of Toronto Institute for Aerospace Studies, Report No. 336, May 1991.

[7]Reid, L.D., Rajagopal, R., and Graf, W.O., *A Simulator Investigation of Helicopter Flight Control System Mode Transitions*. University of Toronto Institute for Aerospace Studies, Report No. 348, January 1994.

[8]Bittner, J.P., *The Use of Texture in Improving the Simulation of Low-Level Flight*, University of Toronto Institute for Aerospace Studies, M.A.Sc. Thesis, 1997.

[9]Shawwa, F.N., *Subjective Comparison of Visual Display Modes for a Nap-of-Earth Mission in a Helicopter Simulator*, University of Toronto Institute for Aerospace Studies, M.A.Sc. Thesis, 1994.

[10]de Sousa, S.B.S., *Analysis of Motion Simulation Capabilities of Reduced Degrees-of-Freedom Flight Simulators*, University of Toronto Institute for Aerospace Studies, M.A.Sc. Thesis, 1998.

[11]Talbot, P.D., Tinling, P.D., Decker, W.A. and Chen, R.T., *A Mathematical Model of a Single Main Rotor Helicopter for Piloted Simulation*, NASA TM 84281, September 1982.

[12]Hoppe, A., *Validation of a Mathematical Model for Piloted Simulation of a Bell 205 Helicopter*, University of Toronto Institute for Aerospace Studies, M.A.Sc. Thesis, 1989.

[13]Reid, L.D. and Nahon, M.A., *Flight Simulation Motion-Base Drive Algorithms: Parts 1,2 and 3*. University of Toronto Institute for Aerospace Studies, Report Nos. 296, 307 and 319, December 1985, May 1986 and December 1986.

[14]Grant, P.R., *Motion Characteristics of the UTIAS Flight Research Simulator*, University of Toronto Institute for Aerospace Studies, TN No 261, 1986.

[15]Kirdeikis, J., *Evaluation of Nonlinear Motion-Drive Algorithms for Flight Simulators*, University of Toronto Institute for Aerospace Studies, TN No. 272, June 1989.

[16]O'Toole, G. J., *An Investigation of the UTIAS Adaptive Washout Algorithm*, University of Toronto Institute for Aerospace Studies, M.A.Sc. Thesis, 1989.

[17]Grant, P.R., *The Development of a Tuning Paradigm for Flight Simulator Motion Drive Algorithms*, University of Toronto Institute for Aerospace Studies, Ph.D. Thesis, 1996.

[18]Sattler, D.E., *Pilot Briefing Document UTIAS E/SVS Experiment 2*. University of Toronto Institute for Aerospace Studies, Flight Simulation Laboratory, Technical Memo, January 1998.

[19]Anon., *NASA-TLX, Task Workload Index*. Human Performance Research Group, NASA Ames Research Center, circa 1988.

[20]Reid, L.D. and Sattler, *Cockpit Technology Simulation Development Study for Enhanced/Synthetic Vision System, Supplemental, Results, Analysis and Discussion - En Route Objective Performance*, University of Toronto Institute for Aerospace Studies, To be Published.

A Design of Computational Environment for the HSE (Human-Sensibility Ergonomics) Simulator

Sugjoon Yoon*
Sejong University, Seoul, Korea

Chul-Jung Kim† and Se-Jin Park†
Korea research Institute of Standards and Science, Daejeon, Korea

Abstract

Human-Sensibility Ergonomics (HSE) is to apprehend human sensitivity features by measuring human senses and developing index tables related to psychology and physiology. One of the main purposes of HSE may be developing human-centered goods, environment and relevant technologies for an improved life quality. In order to achieve the goal, a test bed or a simulator can be a useful tool in controlling and monitoring a physical environment at will. This paper deals with requirements, design concepts, and specifications of the computing environment for the HSE simulator under development, which is a part of HSE Technology Development Program being sponsored by Korean Ministry of Science and Technology. The integrated computing system is composed of real-time and non-real-time environments. The non-real-time development environment comprises several PC's with Windows NT and their graphical user interfaces coded in Microsoft's Visual C++. Each PC independently controls and monitors a thermal or a light or an audio or a video environment. Each of software and database, developed in the non-real-time environment, is directly ported to the real-time environment through a local-area network. Then the real-time computing system, based on the cPCI bus, controls the integrated HSE environment and collects necessary information. The cPCI computing system is composed of a Pentium CPU board and dedicated I/O boards, whose quantities are determined with expandability considered. For the future expansion of the system requirements Firewire real-time communication link is being considered. The integrated computing environment of the HSE simulator is being designed to guarantee

* Member, AIAA.
† Ergonomics Research Group.
Copyright © 1999 by the American Institute of Aeronautics and Astronautics , Inc.
All rights reserved.

real-time capability, stability and expandability of the hardware, and to maximize portability, compatibility, maintainability of its software.

Introduction

Human-Sensibility Ergonomics (HSE) is a technology measuring and evaluating human sensibility qualitatively or quantitatively in order to develop safe, convenient, and comfortable products and environments. This technology will eventually improve life qualities of mankind. HSE, which is to develop human-centered designs for products and environments, consists of three components [1]. The first component is human sensitivity technology to appreciate sensing mechanism by measuring the five senses responding to outer stimuli. The human sensitivity technology is for developing psychology and physiology indices. The second component is simulation technology. This technology is used to apprehend features of human sensory responses. The HSE simulator is a test-bed to test and evaluate psychological and physiological responses to stimuli from products or environments under intentionally controlled physical conditions. The last component of HSE is application technology, which is to apply the HSE concept to designs of products and environments such as automobiles, home electronics, living or working environments, etc..

This paper presents design concepts and requirements of the computing environment for the HSE simulator being developed by KRISS (Korea Research Institute of Standards and Science). The HSE simulator is a testing facility to measure and evaluate human sensitivity factors, such as comfort, satisfaction, movement, spaciousness, and weightiness, processed by a brain and sensory systems including visual, aural, smelling, hearing, and antenna senses as well as an internal ear. For the purpose the simulator sets in a confined space an independent or a composite external environment such as visual scene, sound, smell, taste, temperature, humidity, motion and vibration. When the HSE simulator provides a human with an synthetic environment, the simulator can be used to study human sensitivity technology by measuring human responses to a unit stimulus. The simulator can be also used to develop the application technology, that is, human-friendly goods or environments in the case that composite external environments are furnished. Therefore, the HSE simulator plays a central role among the three HSE components.

In order to design the HSE simulator, experimental facilities in Research Institute of Human Engineering for Quality Life [2] and Life Electronics Research Center in Japan, leading HSE studies in the world, were referenced. Research Institute of Human Engineering for Quality Life in Japan studies, in the HSMAT (Human Sensory Measurement Application Technology) Project, living and working environments yielding less stresses as well as

product designs reflecting human feelings. Life Electronics Research Center in Japan researches physiology such as smelling and visual senses and micro electric currents in the body, as well as the subjects in environmental measurement and control. Especially, the HSMAT Project is very similar to the research program "Development of HSE Technologies" being sponsored by Korea Ministry of Science and Technology, which supports the study in this paper, in a sense that its research objective is to develop human sensitivity and application technologies. However, the research programs in Japan concentrates on HSE technologies in static environments such as living or working spaces while the Korean research program, whose project year runs through 1995 and 2002, is more interested in dynamic environments including automobiles, airplanes, and railway trains. This may be one of major distinguishing points between the Korean and Japanese research programs.

Requirements for the HSE simulator

As mentioned in the introduction, the subject of this paper is a design of the HSE R&D simulator, more specifically its computing environment. Primary simulation objects of the HSE simulator are dynamic environments such as airplanes and automobiles. Based on physiology and psychology index tables obtained from human sensitivity technology, the HSE simulator effectively measures and evaluates responses of human feelings to composite stimuli in a dynamic environment realized by the simulator. The HSE simulator is composed of a control station, a computing system, environmental controlling and sensing devices. The control station allows a researcher to set HSE environments, such as vision, sound, smell, taste, temperature, humidity, and vibration, at will in a confined space with minimum efforts and time. The computing system comprises system S/W and application S/W whose combination evaluates human responses logically in real-time or non-real-time, based on a relevant HSE database. The controlling and sensing devices are to provide composite stimuli in a simulated dynamic environment and measure responses of human sensitivity factors quantitatively. The system configuration of the HSE simulator being developed is illustrated in Fig. 1. In Table 1 are described system requirements relevant to unit testing facilities, each of which generates a different stimulus and measures a relevant human sensitivity factor [1]. Table 1 also shows testing requirements of human sensitivity factors with respect to automobile parts or a whole systems. The requirements are actually adopted by the part suppliers and car manufacturers to test their own products. The HSE simulator integrates testing units, realizes their functions seamlessly in a unified test-bed, and provides synthetic composite environments for the HSE studies.

General requirements applied to the design of the HSE computer system are as follows:

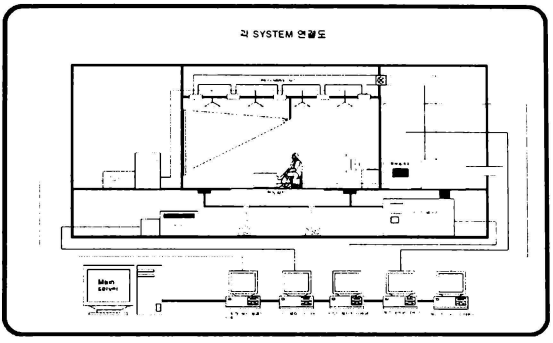


Fig. 1 Configuration of the HSE (Human-Sensitivity Ergonomics) Simulator.

Table 1. Requirement for Evaluation and Application of Human-Sensibility Factors.

Human Sensitivity Factors	Environmental Requirements	Requirements of Car or Parts Manufacturers	
Vision	3D reality of 80% or higher		
Sound and Noise	- air medium: 50~8,000 Hz - solid medium: 40~1,000 Hz - low frequency: 2~50 Hz - when air conditioning stopped: under minimum audible level - when air conditioning operational:: 16dBA	Audio System	100 Hz or less
		Air Conditioner	100 Hz or less
		Seat	100 Hz or less
		Heavy Trucks or Vehicles	10~100 kHz 80 dB or less
		Sedans	125 Hz ~3.2 kHz
Vibration	0.1~50 Hz	Audio System	15~35 Hz
		Air Conditioner	5 kHz or less
		Seat	15~25 kHz
		Sedans	5~400 Hz
Color and Illumination	- lab size: 130~190m^2 variable - transit rate of natural light: 1~5% - brightness of walls and a ceiling: 10~500 cd/m^2		
Humidity	10~95%	Audio System	
		Air Conditioner	10~95%
		Seat	40~95%
		Sedans	
Temperature	- temperature: -15°~+45°C, ±0.5 °C - radiation temperature: ±0.1 °C - air flow rate: ±0.1m/sec	Audio System	
		Air Conditioner	-40°~70°C
		Seat	-30°~150°C
		Heavy Trucks or Vehicles	-40°~100°C
		Sedans	-40°~120°C

- real-time computing capability (update rate: 10hz at minimum)
- stability and durability (operating hours: 8hrs/day, MTBF: 5,000hrs)
- expandability (excessive computing capability and capacity : 50% at minimum)
- S/W portability, compatibility, and maintainability

The above requirements are set based on the experiences and the technical documents [4,5,6,7] in developing airplane, automobile, and railway simulators. Another factor, which has to be considered in the system design, is the operational condition. Research engineers and scientists without any specialties in simulator development or maintenance will operate the HSE simulator. That is, maintenance and upgrade of the computing system will be requested more frequently than in typical training simulators, which are operated for more than 10 years once they are furnished. Therefore, efforts to minimize expenses from maintenance and upgrade should be reflected in the system design. In the sense, required expenses for maintenance and upgrade should be emphasized in the design of the computer system as well as portability, compatibility, and maintainability of the software.

Design of the HSE Computing System

The testing units to study human sensitivity factors were already established in the first development stage (1995 – 1998) of the research program "Development of HSE Technologies". The testing units, reflecting the requirements in Table 1, have been used to study human sensitivity factors, and their technical specifications, which should be considered in the design of the new integrated computing system of the HSE simulator, are summarized in Table 2.

The design requirements of the computing system are categorized into general and HSE-specific ones. First, the previously mentioned general requirements are considered to select computer S/W, H/W, and S/W development tools. In order to satisfy the general requirements as much as possible under budget constraints, Pentium PC's and Windows NT's are selected as distributed computer platforms and their operating systems. As can be seen in Table 2, present computer systems for testing units to study human sensitivity factors are PC's with either Windows NT or Windows 98. That is, time and efforts required for system integration will be very effectively and economically used if PC's and Windows NT's are chosen for the integrated system. In addition to this merit, Windows NT guarantees system stability as well as portability, compatibility, and maintainability of the application software. The remaining problem is that Windows NT is not a real-time operating system. The problem can be resolved by either adopting a real-time kernel or adding real features to Windows NT. To achieve real-time capability

of the system, it should be also considered how to integrate the distributed computer systems.

Methods integrating unit computer systems to study human sensitivity factors are categorized into LAN (Local Area Network) and bus types. When LAN-type integration is decided, Ethernet is usually considered. The relevant communication protocol will be either TCP/IP or UDP/IP. However, TCP/IP, which has a checking function of communication errors, is not generally recommended in real-time environment, because of following non-real-time characteristics:

- Ethernet collisions
- VME bus spin locks (when the server is of VME bus type)
- dropped packets
- timeouts and retransmissions
- coalescing small packets

In the case of UDP/IP, the protocol does not have error-checking functions, even if it allows faster communications than TCP/IP. When UDP/IP is applied to a real-time system and physical distances of communications are not that long, point-to-point type is often chosen rather than net-type communications. In this case, MAC/LLC (Medium Access Control/Link Level Control) layer just above the physical layer should be edited. But the technology requires profound understanding of UDP/IP protocols, which is not available to the development team of the HSE simulator. The typical speed of Ethernet communication is about 10 Mbps. But it is difficult to say that Ethernet communication is slower than bus types since Ethernet products with 100 Mbps are also available these days in the market. Among bus communications, traditional VME bus and recent cPCI bus are two of the most dominant buses in the market. As can be seen in Table 3, cPCI bus is superior to VME bus in system expandability. Both VME bus and cPCI bus satisfy the real-time requirements of the HSE simulator in communication speeds. VME-based computer systems are also affordable because the configuration comprises not many VME cards in this application, even if cPCI system is less expensive. In conclusion, a cPCI-based system has been selected as an integrated computing environment for the HSE simulator, and the expandability of the system is resolved by using Firewire (P1394) [8].

As mentioned before, Windows NT PC's are chosen for the integrated system. The real-time problem of Windows NT can be resolved by either adopting a real-time kernel or adding real features to Windows NT. QNX [9] and Tornado [10], two of the most popular real-time operating systems, provide real-time kernels, while RTX [11] adds real-time capabilities to the NT kernel. They are compared in Table 4. Tornado is the best in real-time capability, but RTX is superior to the others in functions and S/W compatibility.

Table 2 Technical Specifications of Testing Environments for Unit HSE Factors

		Physiology	Riding Comfort and Fatigue	Sound and Vibration
Computer system		PC Pentium II, 450MHz	PC Pentium II, 450MHz	PC Pentium II, 333MHz
Measu-rement system	Sensors	EEG, ECG, EMG, PTG, etc.	3-axis accelerometer, me3000p, biopaq, motion capture system	Microphone, Accelerometer
	Output signals	Digital signal	Voltage	Voltage
	Electric power	110 / 220 V	220 V	220V
	A/D Channels	8 bit, 16 ch	10 bit, 30 ch	10 bit, 64 ch
	Update rate	2 kHz	5 kHz (max)	50kHz (max)
Control system	Controllers			Speakers, Hydraulic valve
	Control signals	Digital signal, Voltage	Voltage	Voltage
	Electric power	110 / 220 V		220V
	D/A Channels	8 bit, 24 ch		10 bit, 64 ch
	Update rate	200 kHz		50 kHz (max)
S/W	Operating system	Windows 98	Windows 98	Windows NT 4.0
	Language	Visual C++, Visual Basic Matlab	Visual C++, Matlab	Visual C++, Matlab
	Tool	MS Office, Lab-view Acknowledge	Lab-view	
	Program/data size	1Mb/3Gb	1Mb/3Gb	1Mb/3Gb

All the real-time capabilities of QNX, Tornado, and RTX are acceptable considering the requirements of the unit facilities to test human sensitivity factors in Tables 1 and 2. Therefore, the first criterion in selecting a real-time operating system (executive) should be their available functions, utilities, and maintainability rather than real-time capability. The prices of QNX and RTX are also about the same and lower than that of Tornado. In conclusion, Windows NT with RTX is selected a real-time operating system for the HSE simulator.

The HSE-specific requirements described in Tables 1 and 2 are reflected in the design of the integrated computing system. The H/W configuration is illustrated in Fig. 2 and the S/W configuration in Fig. 3. The following principles are applied to the system design:

Table 3 Comparison of VME and cPCI buses

	VME	cPCI
Slot #	21	8
Bus speed	80 Mbps (VME 64 ext)	128 Mbps (32 bit) 264 Mbs (64 bit)
# of CPU boards	Multi	Single
Processors	68k, PPC, Pentium, i960	Pentium, PPC
hot swap	No	Yes
Connection	Rugged	Rugged
Bus specification	Verified	Verified
Cost	High	50% lower

Table 4 Real-Time Executives on PC's

		QNX	RTX for Windows NT	Tornado
Operating system		QNX Micro kernel	Windows NT	Wind Micro Kernel
Development tool		Watcom C/C++	Visual C++	GNU C/C++
Debugger			Win DBG/Soft ICE	GNU debugger
Graphic Library		Photon microGUI X Window System for QNX QNX Windows	Any Windows NT compatible graphic libraries	RTX Windows RT GL ZINC
API		QNX dependent API	Win32 API / RTAPI	VxWorks dependent API
RT capa.	Context Switch Time	1.95 μs	4.69 μs	0.9 μs
	Interrupt Latency	4.3 μs	8.78 μs	1.0 μs
	used PC	Pentium, 133MHz	Pentium MMX, 200MHz	Pentium Pro, 150MHz
Cost		< $10,000	$10,000	$22,000

- The HSE simulator is divided into real-time operating and non-real-time development environments.
- The non-real-time environment is composed of PC's with either Windows NT or Windows 98. Their GUI's are programmed in Microsoft's Visual C++. Each PC can independently control the relevant environment such as temperature, humidity, vision, illumination, etc.
- The VME computing system sends on the LAN starting and ending queues of some actions to the VXI system of the motion platform in Figs. 1 and 3, which are the only communication data between two computing systems.
- The VME computing system sends on the bus, starting and ending queues to the physiology measurement system composed of EEG, ECG, EMG, and PTG. The VME computing system intercepts and duplicates the data from the sensors to the processing units of the physiology measurement system.

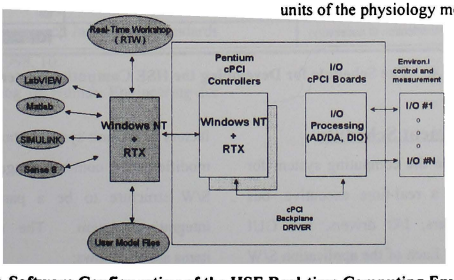
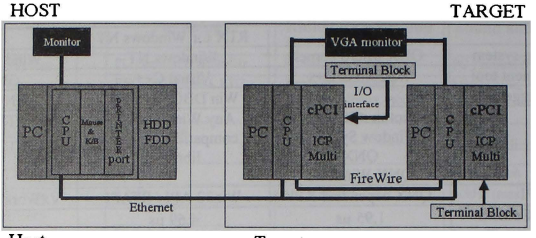


Fig. 2 A Software Configuration of the HSE Real-time Computing Environment.



Host

1. CPU : Pentium II or III
2. Power Supply: 100/200V
3. Mouse & Key board
4. Printer port
5. Windows NT + RTX

Target

1. CPU: ICP-K266-645X
2. Power Supply: 100/200V
3. cPCI rack : 42 TE (Industrial)
4. Windows NT + RTX

Fig. 3 A Hardware Configuration of the HSE Computing System.

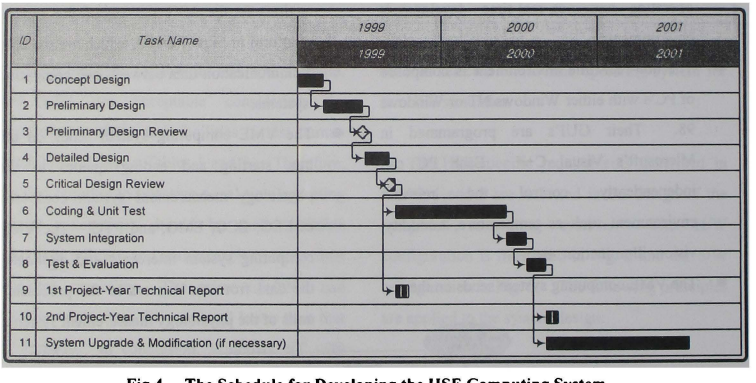


Fig 4. The Schedule for Developing the HSE Computing System.

Development Schedule

In order to realize the computing system for the HSE simulator, a real-time executive, bus communication drivers, I/O drivers, and GUI must be developed. Each of the application S/W for measuring and evaluating human sensitivity factors in the unit systems should be translated or modified in a common language and a unified S/W structure to be a part of the seamless integrating system. The necessary working items are as follows:

- Network Communication
  - control communication between clients and a server
  - edit data received from clients
- cPCI bus driver
  - edit VME bus drivers
  - control communication between Pentium controllers and I/O boards
- I/O device driver
  - edit I/O device drivers
  - edit libraries for I/O device control
- GUI
  - control communication between clients and a server
  - control I/O devices
- Translation of application S/W of Win 32 into RTX

The above working items are being developed as the schedule in Fig. 4.

References

[1] Park S. et al., Development of Simulator Design Technique for Measurement and Evaluation of Human Sensibility, KRISS-98-091-IR, Korea research Institute of Standards and Science, 1998. 10.

[2] "HQL", Catalog of Human Engineering for Quality Life, Japan.

[3] "Life Electronics Research Center", Catalog of Life Electronics research Center, Japan.

[4] Yoon S., Development of a FAA Level 7 Flight Training Device, 3rd year Report, Ministry of Commerce and Trade, 1995.

[5] Yoon S., Bai M., Choi C., and Kim S., "Design of Real-time Scheduler for ChangGong-91 FTD", Proceedings of 'Making it Real' CEAS Symposium on Simulation technologies, Delft Univ., Nov. 1995.

[6] Yoon S., Kim W., and Lee J., "Flight Simulation Efforts in ChangGong-91 Flight Training Device", Proceedings of AIAA Flight Simulation Conference, Aug. 1995.

[7] Yoon S., "A Portability Study on Implementation Technologies by Comparing a Railway Simulator and an Aircraft FTD", Proceedings of AIAA Modeling & Simulation Conference, Aug. 1998.

[8] http://www.inova-computers.com/

[9] http://www.qnx.com/

[10] http://www.wrs.com/

[11] http://www.vci.com/

Simulation for Acquisition, Validation And Electronic System Design

Gary Fitzhugh, Ph.D.
Vice President, Research & Technology
CPU Technology, Inc.
4900 Hopyard Road, Suite
Pleasanton, California 94588

Keywords

VHDL; Verilog; BVT, SystemLab

Abstract

This paper addresses the US Government's thrust in simulation based acquisition. It also addresses hardware description languages that are employed in electronic system simulation. It specifically covers the Behavioral Verification Technology tools employed to verify processor design and SystemLab that is used as an electronic system virtual prototype.

Introduction

The way the US military fights a war has changed, dramatically. Our style of war has become technologically complex and dependent on automated systems. Because of this and because we must find a way to reduce the costs associated with these new technology weapons, modeling and simulation tools will be essential for planning and conducting warfare in the future.

Modeling and simulation must also become a core feature of system design, development, test, and acquisition. With fewer dollars available, the process must become more integrated and more reliant on automated tools rather than the costly empirical methods of the past. This must be done in the design and acquisition process of tomorrow's new technology and in the modernization of legacy systems that will be with us for many more years. It is as the old adage says, "don't work harder, work smarter." Figure 1-1 shows how our acquisition dollars are divided in today's environment.

We need to invest in tools that allow us to work smarter and must less manpower intensive. While less manpower is not popular with many in today's "right sizing" environment, it is none-the-less a fact of life. Much of the total dollar amount associated with the operations and support (O&S) costs of today's weapons systems is in manpower.

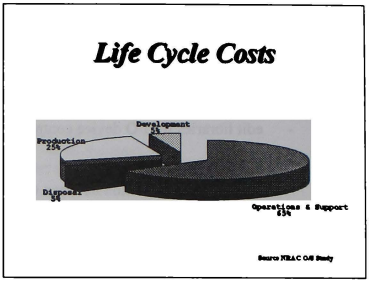


Figure 1-1

There are other costs associated with these O&S outlays, and they too can be attacked through the application of automated tools. It is extremely important that, in the acquisition of new systems, that these issues be addressed in the early stages of the process. What we do at the beginning of the process will affect us during the entire life of the system. Here again we must work smarter and learn from mistakes of the past. Figure 1-2 shows the breakdown of the O&S costs as they stand today. Figure 1-3 show the effects of acquisition process on O&S costs, in milestone segments.

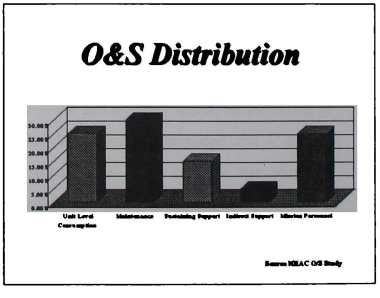


Figure 1-2

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

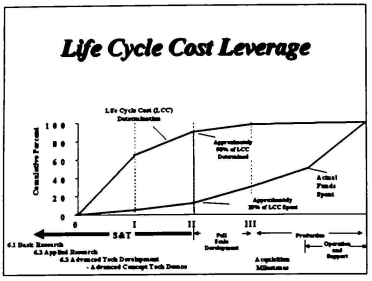


Figure 1-3

Simulation Based Acquisition

In a recent special reprint of The Program Manger, Dr. Patricia Sanders, Director Defense Test, System Engineering and Evaluation, indicated that the Department of Defense (DoD) envisions an acquisition process supported by the robust, collaborative use of simulation technology that will be integrated across acquisition phases and programs. She went on to say that the objectives of DoD's Simulation Based Acquisition (SBA) initiative are being designed to:

- Reduce the time, resources, and risk associated with the acquisition process
- Increase the quality, military utility, and supportability of systems developed and fielded
- Enable integrated product and process development from requirements definition and initial concept development through testing, manufacturing, and fielding

While we are aware of the many diverse simulation programs that exist within the military-industrial complex, it is unclear what standard will be (or can be) mandated for SBA.

In the following paragraphs, we will talk about the various standards that are used in industry and specifically address automated design tools that are used by CPU Technology Inc.

Electronic Design Modeling

VHSIC Hardware Description Language (VHDL)

VHDL is designed to fill a number of needs in the design process. First, it allows description of the structure of a design. In other words, how it is decomposed into sub-designs, and how those sub-designs are interconnected. Second, it allows the specification of the function of designs using familiar programming language forms. Third, as a result, it allows a design to be simulated before being manufactured, so that designers can quickly compare alternatives and test for correctness without the delay and expense of hardware prototyping.

In September of 1996, the DoD published Mil-Hdbk-62. The stated purpose of this handbook is: "This handbook was developed to provide guidance to Department of Defense personnel who are writing requests for proposals for military digital electronic systems, DoD contractors who are VHDL models for the Government, and DoD engineers, scientists, and management or independent validation and verification contractors who are evaluating or reviewing models delivered to the Government." It documents the state of the art and existing technologies for VHDL model development. Addressed in the handbook are the specific VHDL models required to be delivered with a contract, the VHDL models that should be developed during the different stages of the lifetime of a system, and how VHDL models can be structured to be consistent with modeling standards.

Now one could say that this is a rather strong push for a standard for SBA. However, if one reads a bit further, this same document also states, "This handbook is for guidance only. This handbook cannot be cited as a requirement. If it is, the contractor does not have to comply."

VHDL is a language used to describe digital electronic systems. It arose out of the United States Government's Very High Speed Integrated Circuits (VHSIC) program, initiated in 1980. In the course of this program, it became clear that there was a need for a standard language for describing the structure and function of integrated circuits (ICs). Hence the VHDL was developed, and subsequently adopted as a standard by the Institute of Electrical and Electronic Engineers (IEEE) in the United States.

Verilog

Verilog is one of the two major Hardware Description Languages used by hardware designers in industry and academia. VHDL is the other one. The industry is currently split on which is better. Many feel that Verilog is easier to learn and use than VHDL. VHDL was made an IEEE Standard in 1987, and Verilog in

1995. Verilog is very C-like and liked by electrical and computer engineers as most learn the C language in college. VHDL is very Ada-like and most engineers have no experience with Ada.

Verilog was introduced in 1985 by Gateway Design System Corporation, now a part of Cadence Design Systems, Inc.'s Systems Division. Until May, 1990, with the formation of Open Verilog International. Verilog HDL was a proprietary language of Cadence. Cadence was motivated to open the language to the Public Domain with the expectation that the market for Verilog HDL-related software products would grow more rapidly with broader acceptance of the language. Cadence realized that Verilog HDL users wanted other software and service companies to embrace the language and develop Verilog-supported design tools.

Verilog HDL allows a hardware designer to describe designs at a high level of abstraction such as at the architectural or behavioral level as well as the lower implementation levels (i. e. , gate and switch levels) leading to Very Large Scale Integration Integrated Circuits layouts and chip fabrication. A primary use of HDLs is the simulation of designs before the designer must commit to fabrication.

Industry Requirements and Developments

Designers, manufacturers, integrators and users have different validation objectives. Of course, each of these groups wants the final device to have no bugs, but their direct involvement is limited.

Designers need to verify each design element and get the processor to a level of functionality to prove performance assumptions as quickly as possible, so they can move on to the next design.

Manufacturers need to extend the architecture to open up new applications and to ensure compatibility to some acceptable criteria as quickly as possible so that the product can be shipped.

Systems companies and device integrators want consistent testing applied by all manufacturers to provide verifiably compatible devices against standard criteria.

End-users want as much testing done at every level as is possible. They also want low price, high performance, new features and no problems with their applications, even future ones.

There is a team that is believed to be working on the specification, simulation and verification of high-level system-on-chip descriptions via extensions to the Verilog language. The goal of this team is much earlier specification and verification than is currently possible. Verilog, as is, is too low (as a simulator) and too slow. While VHDL is an excellent simulation tool, it has its own set of problems when used for synthesis. So what is the answer?

The ideal solution to the verification problem would embody the following characteristics:

- The validation process would be automated, eliminating the uncertainty of ad-hoc, random testing.
- The process would be repeatable.
- The testing would be concise and would identify what works and what does not.
- The process could be used in all phases of processor design, from architectural analysis to finished silicon validation.
- The process would be easy to use.

Behavioral Verification Technology™ (BVT)

Understanding that this paper is focused on the design of electronic systems, it is also important to acknowledge that most such systems are processor based. BVT, which is detailed below, is best defined as an executable specification. Figure 1-4 shows this relationship.

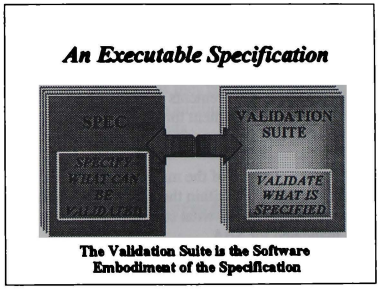


Figure 1-4

CPU Technology's Behavioral Verification Technology consists of an extensive set of software programs. Each program is comprised of a set of self-diagnosing tests. The expected results of each test are embodied in the test program itself. These pre-engineered expected results are created in a semi-

automated method utilizing custom development tools. As the software program performs diagnostic tests on the processor, any discrepancy between actual and expected results is immediately reported.

All specified behaviors of each processor function are thoroughly tested. Each test case is composed of code which sets up the initial state of the processor, performs the far call instruction, and then compares the actual results against the expected results. Each test checks that proper results are stored in registers, condition codes, and memory. In addition, tests are performed to detect undesired side-effects of the processor function. Figure 1-5 shows the effects of a properly verified processor.

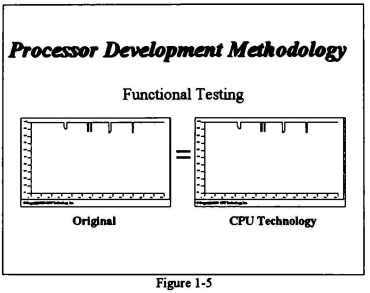


Figure 1-5

Behavioral Verification Technology can be applied in both pre-silicon and post-silicon environments.

Processors are the most complex building blocks ever made by man. Transistor counts in processors are doubling every two to three years. Because functional complexity can double with only an incremental increase in transistor count, this complexity, as well as probability of design errors, is growing at an exponential rate. Traditional methods of testing cannot adequately verify the correctness of these devices. CPU Technology's Behavioral Verification Technology provides a thorough, effective solution to the problem of validating the functionality of increasingly complex processors.

SystemLab™

Most simulators in use today for digital systems design have only the capacity to handle a single ASIC device or perhaps a few devices. This limitation imposes a piecemeal approach to a design since the actual system debug is delayed until the physical prototype is built. This delay and the inefficiencies associated with it increase the development cycle. Also, designers must create and debug sets of test vectors which is very time consuming. Bus functional models that generate cycles on a processor bus are used to produce a limited number of test scenarios in an attempt to alleviate part of the problem. But any actual software that is to be used to test the functions must first be mapped into commands to drive the bus functional models. Simulations typically run at about five to twenty clocks per second, so the amount of testing and validation that can be accomplished prior to building physical devices is severely hampered.

Another approach is to run the test software on the host processor using a software emulator. However, it is not possible to obtain completely accurate timing information from the resulting simulation. Similarly, hardware modeling -- connecting an actual physical device into the simulation environment on which to execute code, is also used by designers. Hardware models are, by definition, cycle accurate, but they suffer from several limitations. They are in effect "black boxes" in which there is no ability to observe or manipulate their internal state. Also, the hardware model can only exist if the processor itself exists. This approach cannot be applied to devices that are still on the drawing board.

When debugging physical prototypes, or emulators, our visibility is limited to the pins on a chip that can be probed, and is further limited by physical design constraints. This is becoming more critical as advanced microprocessors with large internal caches proliferate and, in general, as chips become more highly integrated. Is a signal that we want to observe even available to be probed? And if so, is there room to attach a logic analyzer probe on a signal? In the case of emulators, the designer can re-route the board in order to move important internal signals to pins, but this can take weeks for each iteration.

CPU Technology's approach, SystemLab™, expands beyond hardware/software co-design to the virtual laboratory which consists of virtual prototypes made from software models rather than hardware components. Figure 1-6 is a SystemLab screen.

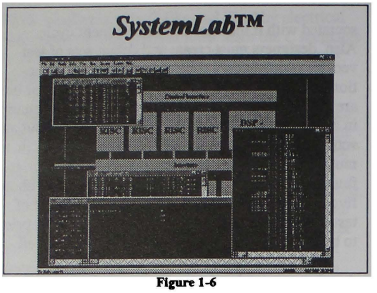


Figure 1-6

Figure 1-7 shows many of the SystemLab features.

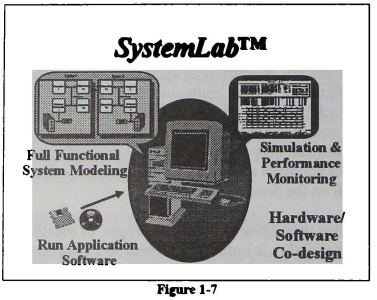


Figure 1-7

SystemLab also includes equipment normally found in a well-equipped laboratory, such as an in-circuit emulator, logic analyzer, oscilloscope, and clock generator, all in virtual form. Software tools such as assemblers and disassemblers are also integrated into the environment, as well as a level of instrumentation and visibility not available in the physical world. These capabilities combine to provide accurate, full system simulation, rapid analysis and debug which can be applied to new and legacy systems. Figure 1-8 shows the imbedded laboratory environment available in SystemLab.

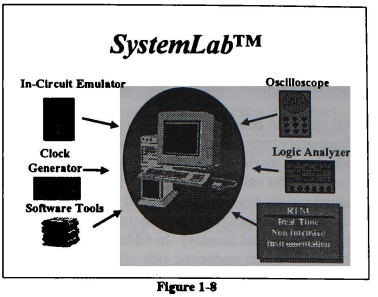


Figure 1-8

References

Department of Defense. 1996. Documentation of Digital Electronic Systems with VHDL. Defense Supply Center Columbus, ATTN: Director-VA, 3990 East Broad Street, Columbus, OH 43216-5000 (Sep).

Sanders, P. 1997. "Simulation Based Acquisition: An Effective, Affordable Mechanism for Fielding Complex Technologies." In Program Manager Special Reprint. Washington, DC (Sep).

Gary L. Fitzhugh, Ph.D.

Dr. Fitzhugh is one the pioneers in electronic systems design simulation and has been active in the microelectronics engineering field for over 20 years. He currently serves as a CPU Technology Vice President, responsible for the introduction and pursuit of emerging technologies. Prior to this position, Dr. Fitzhugh served in several key positions in the both Defense Industry and the US Government. Dr. Fitzhugh holds a Ph.D. in Computer Engineering Science, a MBA in Finance and Accounting, and a BA in Math/Physics.

IMAGE DYNAMIC MEASUREMENT SYSTEM (IDMS-2) FOR FLIGHT SIMULATION FIDELITY VERIFICATION

R. D. Lehmer and W. W. Y. Chung
Logicon Information Systems and Services
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

The fidelity of human-in-the-loop flight simulators is critically dependent upon maintaining a short time interval in the response of the motion and visual cues to a change in the simulated aircraft response. The delay in the visual and motion cues is of particular concern in research and development flight simulators because the pilot perceives the simulated aircraft response and performance primarily through these simulation cues. While the delay in the motion base response can be monitored with accelerometers and other sensors, the measurement of the delay in the visual system response time requires an instrument for analyzing the video data in real-time. The Image Dynamic Measurement System, Mark II (IDMS-2) has been developed at NASA's Ames Research Center to analyze the output video of an image generator and to produce analog and digital outputs proportional to the position of a predefined target image. The IDMS-2 is designed to support a wide variety of video sources and scan rates and has an adaptive timing system that allows for analysis of the target image during each video field, independent of scan rate or interlace factor. In addition, the logic contained within the IDMS-2 prevents erroneous data from being generated if the instrument is not configured properly for the video source being analyzed. In this paper, the design and implementation of the IDMS-2 instrument will be presented. The use of the IDMS-2 in verifying the visual cueing fidelity of the Vertical Motion Simulator at Ames Research Center will also be discussed.

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U. S. Code. The U. S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner.

Introduction

In order to quantify the perceived response of a simulated aircraft based on the simulator's visual system, it is necessary to measure the visual system's throughput delay characteristics by electronically

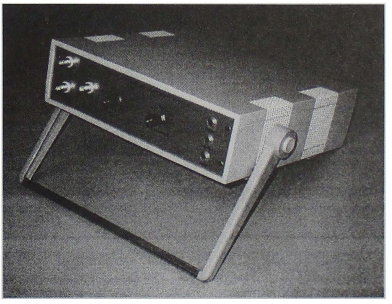


Figure 1: The portable version of the IDMS-2 developed by Logicon Information Systems and Services for NASA's Vertical Motion Simulator facility at Ames Research Center.

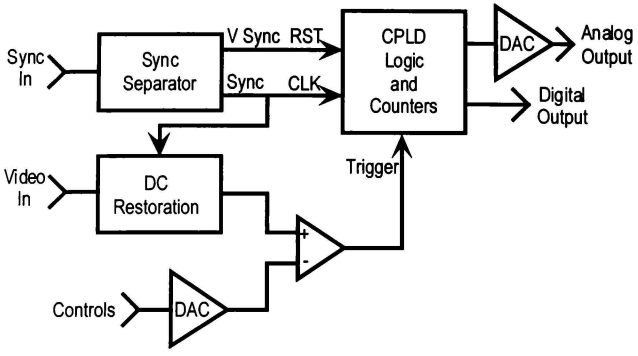


Figure 2: Schematic diagram of the functional blocks that make up the IDMS-2 instrument. Both the threshold voltage generator and the output stage gain are controlled via front panel controls.

tracking the motion of visual targets. At NASA's Ames Research Center, the Image Dynamic Measurement System, Mark II (IDMS-2) has been developed as an instrument for determining the position of these visual targets. In conjunction with a cross-correlation analysis technique, the delay in the generation of the visual targets can be compared to the visual system command signal in the mathematical model or other cueing devices. The instrument has been developed in both rack-mount and portable packages (Fig. 1).

Instrument Description

The IDMS-2 is designed to analyze NTSC and RGB video signals and generate analog and digital representations of the position of a target. The target for this instrument is one-dimensional in that it is made up of a transition during a vertical scan from a dark field to a light field. The IDMS-2 has the flexibility to handle different types of video sources and thus has separate inputs for video and sync signals in order to accommodate RGB video systems with separate sync signals. NTSC video or RGB video with sync-on-green have composite video sync and data on the same channel, and the video signal should be fed to both inputs[1]. To allow the IDMS-2 to be used with a variety of coaxial cables, the inputs of the instrument have high impedance. Each input signal cable can then be externally terminated into its characteristic impedance as required by the particular application.

The IDMS-2 consists of several functional blocks, including a sync separation circuit, a video discrimination circuit, and an analog and digital output processor (Fig. 2). While the functional blocks that handle analog signals use discrete components, all of the digital logic is contained in a Complex Programmable Logic Device, or CPLD[2]. The Xilinx XC9500[3] series CPLD was selected for this instrument because it provides excellent flexibility due to its In-System Programming feature[4]. Since the IDMS-2 is a research instrument, the ability to reprogram the instrument to meet future requirements is highly valuable. However, the IDMS-2 has been designed not to require any user programming for normal operation.

The sync separation circuit generates separate signals for vertical and horizontal sync from the signal applied to the Sync In input. The sync separation circuit

has a horizontal line scan rate in excess of 150kHz, and thus has sufficient bandwidth to process video signals from high-resolution image generators. The IDMS-2 uses the vertical sync to reset a set of horizontal line counters that are clocked by the horizontal sync signal. When a dark to bright transition is detected by the video discrimination circuit, the current horizontal line count is latched to a DAC for the analog output and a separate set of latching buffers for the digital output.

The video discrimination circuit has several components, including a DC restoration circuit, a programmable threshold voltage, and a comparator. The DC restoration circuit detects the "dark" level at the beginning of each line of video data. The comparator stage compares the difference between the video level relative to the "dark" level and the user selected threshold level. The threshold level is set by a DAC controlled by a sixteen position rotary switch on the front panel of the instrument. When the video data exceeds the threshold voltage within any video line, the comparator sends a trigger signal to the output processor.

The output processor stage relies on a 12-bit digital counter driven by the horizontal sync signal to count the number of horizontal lines since the last vertical sync signal was asserted. When the comparator in the video discrimination circuit observes a transition, the trigger from the comparator fires a one-shot that clocks the current contents of the counter into a set of latches for the 12 bit digital output.

The analog output is derived from a 12 bit latching DAC. The output range is specified in terms of producing a 10 Volt maximum output for a range of lines per field: 4096, 2048, 1024, 512, and 256. For each step in this range, the value of the digital counter is digitally multiplied by 2. Thus, the dynamic data of the counter will be in the top bits of the DAC. Therefore, taking the 256 lines per field setting as an example, only the lower 8 bits of the counter will be significant. In order to generate the 10 Volt output on the DAC, these 8 bits of data have to be shifted to the top 8 bits of the DAC. Since this operation is done digitally, a separate output stage with programmable gain is not necessary.

The IDMS-2 processes each field of video data presented, independent of whether the field is part of an interlaced or non-interlaced frame (a frame is one complete video picture). To eliminate any confusion in determining the location of the target during a scan of a video field, only one transition is allowed per field. The one-shot that controls the latches is not reset until a vertical reset is generated by the sync separation circuit at the end of the video field. Therefore, the bandwidth for updating the position data is the video field rate. For example, if an image generator is producing 30 frames per second and presents each frame as two interlaced fields, position data generated by the IDMS-2 will be updated at 60 Hz. In addition, the digital and analog outputs are updated immediately upon detection of a transition and are held constant until the next transition is detected in a subsequent field.

Since the target detection is based upon observing the brightness of the video signal, certain types of video signals are not compatible with the IDMS-2. As a specific example, the IDMS-2 can not properly process NTSC signals that have closed captioning data embedded in the first few lines of each field. In the case of closed captioning, the embedded data has the same amplitude as the maximum brightness level for NTSC video, thus generating a false trigger early in each vertical field scan. The IDMS-2 is also not equipped to process video signals with digital sync signals, such as VGA or serial digital video.

Operation

To configure the IDMS-2 for normal operation, the video signal to be analyzed is first connected to the inputs of the IDMS-2. Depending upon the configuration of the flight simulation facility, the connection to the IDMS-2 can be accomplished by splicing into the existing lines to the cockpit display units or replicating the video signal with either a distribution amplifier or a video matrix switch (Fig. 3). The output of the IDMS-2 is connected to a data acquisition computer that also monitors the command signal to move the target on the image generator. Again depending upon the configuration of the simulator facility, the data acquisition system could either be the

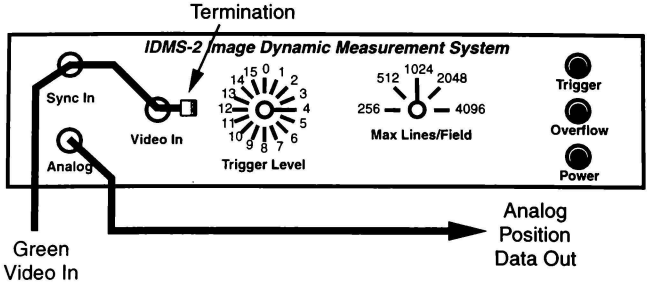


Figure 3: Typical connection diagram for the IDMS-2 instrument when analyzing a "sync-on-green" RGB video signal source.

host computer that operates the simulator or a separate piece of dedicated data acquisition hardware.

Once the IDMS-2 is physically connected, the user then configures the threshold level and number of field lines to process. The front panel Trigger Level selector allows for the choice of one of sixteen linearly increasing threshold levels. Valid transitions are best discriminated from the background dark level and other noise sources by selecting the highest threshold level that is still less than the video level of the bright portion of the field. If the behavior of the threshold detector needs to be checked, the two inputs to the internal comparator of the IDMS-2 are available as test points on the rear of the instrument. With these test points, the proper operation of the threshold voltage DAC and the DC restoration circuit can be checked, as well as letting the user monitor the input video data without disturbing the input connections to the instrument.

In addition, the Trigger lamp on the front of the IDMS-2 instrument can be used as a visual indication that the transitions are being detected in each field displayed. If the Trigger lamp flashes irregularly, it is an indication that the threshold level may be too high, or that the transitions are not being generated in each field.

The other front panel indicator is the Overflow lamp. The overflow indication can result from one of two conditions associated with the digital data for the output DAC overflowing. In either case, the number of lines in the video field has to be less than the maximum number of lines selected on the front panel selector in order for the IDMS-2 to generate an Overflow indication. In the first case, the transition in the video level occurs at a line greater than the maximum number selected, while in the second case, no transition occurs at all because of an all black field, for example. However, in the case of an all black field, no Trigger indication will be seen. In the event of an overflow condition, the DAC output is set to maximum (10 volts). The maximum output level is held until the next valid trigger is observed in a subsequent field. The overflow indication has no effect on the digital output since it has 12-bit resolution and simply counts the number of lines since the last vertical reset to the video transition.

Discussion

The IDMS-2 has been fully integrated and tested at the Vertical Motion Simulator (VMS)[5]. The VMS is the world's largest flight simulator motion base, with a vertical displacement of 60 feet and a lateral displacement of 40 feet (Fig. 4). The out-the-window

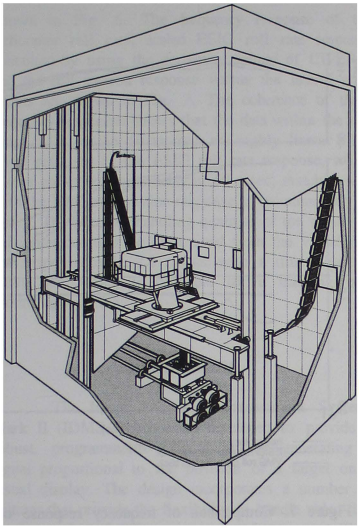


Figure 4: Cutaway drawing of NASA's Vertical Motion Simulator (VMS) tower.

visual scenes are generated by one of two Evans and Sutherland image generators (an ESIG-3000 or an ESIG-4530) that are driven via a dedicated Ethernet network by a real-time executive running on Compaq Alpha processors. The real-time executive manages the math model of the simulated aircraft, the I/O for the motion base and cockpit instrumentation, and the collection and recording of various model parameters during a simulation.

In addition to the image generators for the out-the-window scene, several SGI workstations are also available to generate HUD (head's-up displays) and HDD (head's-down display) graphics and are also connected to the real-time host via the same dedicated Ethernet network. All of the video sources for the laboratory are selectable through a video switch matrix

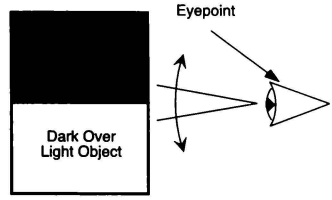


Figure 5: Schematic representation of the relative positions of the side viewing eyepoint on the simulated aircraft and the target object used for a visual throughput test.

and can be processed and mixed for a variety of researcher and pilot applications.

Since the IDMS-2 is to be used to determine the total delay in the generation of visual cues from changes in simulated aircraft position and attitude, the video target for the instrument has to be generated using the standard out-the-window scene configuration of the image generator. The only special requirement in the image generator configuration is the placement of a tower-like object in the image generator's database that has a dark top and a white bottom (Fig. 5). The aircraft's position and attitude is set so that the eyepoint is directly in front of the tower object and is so close to the tower that it completely fills the field of view in both the horizontal and vertical directions. The actual axis to be tested depends upon the requirements of the simulator fidelity test, but typically vertical and pitch axes are tested using the forward looking eyepoint of the center window of the image generator, while roll and lateral axes can be tested using a side viewing eyepoint.

For the characterization discussed in this paper, a generic helicopter model is used and the fidelity of the cues for the roll axis is studied by driving the motion base and visual system with the real-time helicopter model solely along the axis in question. In order to factor in the total delay in the presentation of the visual cues, the video is monitored at the last distribution amplifier before entering the cockpit. Thus, the delay in the visual cue will include the host computer, the real-

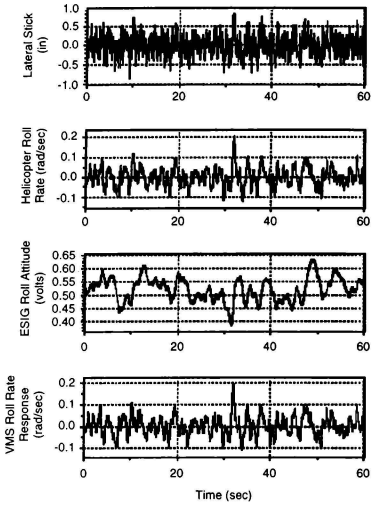


Figure 6: Time histories of the raw data from a roll rate fidelity study for the VMS. Shown are the lateral input command signal and the corresponding roll behavior computed in the model and measured in both the visual and motion systems.

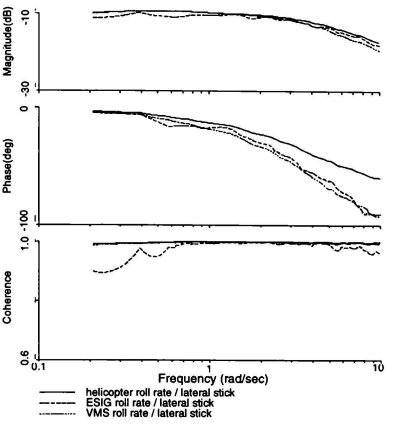


Figure 7: Comparison of frequency response of various roll rates to lateral stick input command. Note that preceived roll rates from the visual and motion cues are quite similar in this characterization test of the VMS.

time network, the image generator, all of the video distribution, and the video switch matrix.

The Evans and Sutherland ESIG-3000 image generator used in these tests generates RGB video with sync information on the green channel. Thus, the green video signal is used as the input to the IDMS-2 and the analog output of the instrument is connected to an analog to digital converter (ADC) channel in the real-time I/O system. The ADC channel is sampled during each cycle of the real-time system, which has a maximum frame rate in excess of 100 Hz, depending upon simulation complexity and operational requirements.

To determine the simulated aircraft response characteristics, the perceived visual cues are measured by the IDMS-2 and the effective image response time of the ESIG-3000 is measured. In addition, measurements of the perceived motion cues of the VMS motion base are made simultaneously for comparison with the visual cue data. A white noise source with a Gaussian distribution of the energy over the frequency spectrum is fed into the lateral stick input of a helicopter with a roll damping characteristics of 4 rad/sec in a roll rate command system. The time history of the white noise control inputs, helicopter roll rate, roll attitude video measurement, and the roll rate motion response are

shown in Fig. 6. The frequency response of the helicopter roll rate, scaled ESIG roll rate response (obtained by using the derivative feature of CIFER[6]), and the roll motion response versus the lateral stick inputs are shown in Fig. 7. The coherence of these frequency responses shows that the data within the test range, i. e., up to 10 rad/sec, are highly linear. The phase response of the ESIG roll rate response, which shows an approximate delay of 60 msec, matches well with the VMS roll motion response as recommended by Ref. 7. The effective handling qualities of the simulated helicopter can then be determined from the visual roll rate response to be 2.8 rad/sec with a time delay of 0.046 sec, which is desirable for a tracking task according to Ref. 8.

Conclusions

The Image Dynamic Measurement System, Mark II (IDMS-2) has been developed to provide a robust, programmable instrument for generating a signal proportional to the position of a target on a visual display. The design incorporates a number of features that provide the flexibility to handle video data from a number of different image generators, as well as minimizing the number of field adjustments necessary to keep the instrument operating properly. In addition, the use of a white noise distribution for the lateral command signal to the helicopter model and the corresponding roll response generated by the visual system allows for a thorough analysis of the frequency response and delay of the visual cueing system of a flight simulator. Both the motion and visual systems of the Vertical Motion Simulator have demonstrated similar delays in cue response. However, the performance of these systems is a function of the complexity of the simulation, thus a flexible and adaptable instrument such as the IDMS-2 reduces the difficulty in determining the fidelity of the visual cues in a particular simulation.

Acknowledgments

The authors would like to thank the engineers and technicians of the Engineering and Operations Groups of Logicon Information Systems and Services, Moffett Field, CA for their help developing the IDMS-2 instruments. In particular, the authors wish to thank Mr. Soren LaForce, Mr. Irwin Tio, and Mr. Kevin Chargin of Logicon Information Systems and Services, and Mr. Duc Tran of NASA Ames Research Center. Finally, we would like to acknowledge the work of the late Mr. Carl Roach in developing the original IDMS for the Vertical Motion Simulator. This work was conducted as part of NASA contract NAS2-13868.

References

[1] Jack, Keith, *Video Demystified, a Handbook for the Digital Engineer, 2nd. Ed.*, HighText Interactive, Inc., San Diego (1996).

[2] Sharma, Ashok K., *Programmable Logic Handbook*, McGraw Hill, New York (1998).

[3] Xilinx and XC9500 are registered trademarks of Xilinx, Inc.

[4] *The Programmable Logic Data Book*, Xilinx, Inc., San Jose (1998).

[5] *Vertical Motion Simulator Familiarization Guide*, National Aeronautics and Space Administration, Simulation Experiments Branch, Ames Research Center, Moffett Field, CA, October, 1991.

[6] Tischler, M. B.; and Cauffman, M. G., "Frequency Response Method for Rotorcraft System Identification: Flight Applications to the BO-105 Coupled Rotor/Fuselage Dynamics." Journal of American Helicopter Society, Vol. 37, No. 3, pp. 3-17, July, 1992.

[7] Chung, W.Y. and Schroeder, J.A., "Visual and Roll-Lateral Motion Cueing Synchronization Requirements for Motion-Based Flight Simulations," American Helicopter Society's 53rd Annual Forum, April, 1997.

[8] *Aeronautical Design Standard, Handling Qualities Requirements for Military Rotorcraft*, ADS-33D, July, 1994.

DETERMINATION OF NAVION STABILITY AND CONTROL DERIVATIVES USING FREQUENCY-DOMAIN TECHNIQUES

William D. Lewis, PhD
Associate Professor
AIAA Member

R. Charles Catterall
Graduate Research Assistant
AIAA Student Member

Aviation Systems and Flight Research Department
The University of Tennessee Space Institute
Tullahoma, TN

Abstract

Frequency-domain systems identification techniques were used to extract the longitudinal and lateral/directional stability and control derivatives of a Ryan Navion, a general aviation type aircraft, from flight data. The extracted/identified stability and control derivatives were compared to wind-tunnel results, as well as time history plots of the model versus flight data. The results of the study showed that the identified derivatives closely matched (in magnitude and phase) the aircraft's dynamic response to control doublets. Moreover, the identification was carried out successfully for the Navion using an "inexpensive" instrumentation system comprised of an analog-to-digital converter, a laptop computer and conventional flight test sensors.

Symbols

AX	longitudinal acceleration (a_x), ft/sec^2
AY	lateral acceleration (a_y), ft/sec^2
AZ	normal acceleration (a_z), ft/sec^2
DAIL	aileron surface deflection (δ_a), deg
DELE	elevator surface deflection (δ_e), deg
DRUD	rudder surface deflection (δ_r), deg
P	roll rate (p), deg/sec
PHI	roll attitude (ϕ), deg
Q	pitch rate (q), deg/sec
R	yaw rate (r), deg/sec
THTA	pitch attitude (θ), deg
U	longitudinal speed component (u), ft/sec
V	lateral speed component (v), ft/sec
W	normal speed component (w), ft/sec

All other notation conforms to the convention outlined in Reference 1.

Copyright © 1999 by R. Charles Catterall. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

Introduction

This study was a part of the ongoing frequency-domain flight testing by the University of Tennessee Space Institute Aviation Systems and Flight Research Department. The study involved the planning and execution of flight tests of the Ryan Navion for the identification of the lateral/directional and longitudinal stability & control derivatives using frequency-domain systems identification methods. The purpose of the study was twofold: (1) to apply frequency domain identification techniques, with minimal investments in measurement equipment, to a general aviation aircraft; and, (2) to identify an aircraft model to support future frequency-domain research activities involving the Navion (N66UT).

Originally, the intent of the study was to develop a full 6 degree-of-freedom (DOF) rigid-body model, including any thrust coefficients. However, the initial model identification proved that the aircraft's longitudinal and lateral/directional dynamics were decoupled. In addition, the thrust coefficients were dropped from the identification process due to the highly nonlinear behavior of the thrust/throttle control system. Consequently, 3 degree-of-freedom models for the lateral/directional and longitudinal dynamics of the aircraft were identified and verified.

Background

State-space modeling provides a characterization of the [coupled] aircraft dynamics in terms of ordinary linear differential equations with constant coefficients. The coefficients of these equations are the force and moment (stability and control) derivatives of the equations of motion as commonly derived using small-perturbation theory.

System identification, as applied to state-space modeling, is a procedure for accurately characterizing the dynamic response of a complete aircraft, subsystem, or individual component from measured data.[2] This technique eliminates the need for comprehensive, high fidelity analytical modeling with its inherent assumptions. Typical applications involving system identification include:

- simulation model development
- comparison of wind tunnel and flight characteristics
- control system design and optimization

Navion (N66UT) Research Aircraft

Aircraft Description

The research aircraft is a Navion (N66UT) manufactured by Ryan Aeronautical Company of San Diego, California. It is a low-wing, four-place airplane powered by a single air-cooled engine with tricycle landing gear. Its fuselage is an all-metal one-piece semi-monocoque structure; the tail unit is a cantilever monoplane type with detachable tips. A three-view drawing of the Navion is presented in Figure 1. A detailed description of the Navion, including aircraft specifications, may be found in Reference 3.

The Navion (N66UT) is owned and operated by the University of Tennessee Space Institute, Tullahoma, Tennessee. It is currently used in its capacity as an in-flight simulator to aid in aircraft stability and control instruction, as well as a research aircraft for flight test projects conducted by the Aviation Systems and Flight Research Department.

Aircraft Modifications

The Navion (N66UT) has several modifications made by Princeton University under various United States Navy grants that contrast it from a production Navion. The most notable modification to the N66UT is an analog variable-stability (variable-response) computer and fly-by-wire (FBW) control system. The FBW system, commanded from the pilot's station, is comprised of an assortment of sensors, hydraulically actuated control surfaces (commanded by electrical signals), and cockpit controls that are integrated by the analog computer. The computer, housed within cabin in place of the two rear seats, allows the pilot to modulate the system's control and sensor signals to achieve an airplane response of a particular character and magnitude.

Other modifications to the N66UT, not part of the FBW system, include:

1. The Continental 185 hp engine is replaced by a Teledyne-Continental IO-520B, rated at 285 take-off horsepower (2700 RPM at sea level).

2. A McCauley three-bladed variable pitch propeller (part number D3663/582NC-2) is installed in place of a two-bladed propeller.

3. The main landing gear struts are replaced with those designed for the Camair twin (a Navion conversion with nearly 40% increase in gross weight).[4] Additionally, the landing gear is fixed in an extended position, and the hydraulic control extension/retraction controls are disabled.

4. The flap system has hinged and actuation modifications that allow the flaps to be deflected up, as well as down, within a ±30 degree range.

5. The mechanical rudder-to-aileron coupling system (standard on all production aircraft), that applies coordinated aileron movement for pedal input, is replace by conventional rudder controls.

6. The vertical tail surface area is increased by means of a chordwise extension and a spanwise cap on the rudder surface.

Flight Test

Flight Test Conditions

The short-term frequency response flight tests of the research aircraft, a Ryan Navion (N66UT), consisted of a data collection flight followed by a validation flight of 1.5 and 0.7 hours, respectively. The flight tests were flown in calm air under daylight visual meteorological conditions (VMC) at the Tullahoma Regional Airport, Tullahoma, Tennessee, on the 9th and

11th of October 1998. Trim configuration was steady-state horizontal flight at 90 KCAS at 5000 ft density altitude (H_d); the gross weight (GW) and center of gravity (c.g.) for the data collection flight were 3,242 lbs and 100.3 inches aft of datum [24.4% mean aerodynamic cord (MAC)], respectively. Furthermore, the tests were conducted within the existing envelope of the aircraft with the canopy closed, flaps at 0° and landing gear extended with the heater and carburetor heat off. The essential test elements for the data collection flight included three on-axis pilot-generated, frequency-sweep inputs and up to four doublet inputs for each control axis (directional, lateral, longitudinal, and thrust/throttle). The validation flight was performed to observe the aircraft's dynamic characteristics, including the phugoid and Dutch roll, at the test condition.

Flight Test Method

The flight test technique to acquire data consisted of pilot-generated "frequency-sweeps" using each on-axis control [independently] to excite the vehicle dynamics over a broad frequency range. A typical sweep (see Figure 2) was initiated with two low-frequency sinusoid-shaped cycles (about the trim flight condition) with a period of 20 seconds, corresponding to the lower end of the desired identification frequency range. The sweep frequency was then increased exponentially to the highest frequency the pilot could generate (approximately 2 Hz). At the same time, the control displacements are increased/decreased to effect a noticeable aircraft response, but not increased so large as to generate large airspeed changes (greater than ±10 knots), translations or attitudes -- or risk damaging the aircraft. In addition to sweeps, doublets were performed, again using each on-axis control, to provide dissimilar data for model verification purposes.

Data Collection

Data were collected at 100 samples/second using a 16 channel, 12-bit A/D converter (without filters) and a 120 Mhz Pentium® laptop computer wired to conventional flight test sensors, including rate gyros, accelerometers, and potentiometers. The computer recorded all data for post-flight retrieval and displayed real-time strip charts of each channel for data monitoring. The data shown in Table 1 were collected for identification analysis.

Identification

Data Conditioning and Analysis

Analysis of the data was conducted using CIFER® (Comprehensive Identification from FrEquency Responses), a systems identification software facility developed by the U.S. Army/NASA and Sterling Software. The foundation of the data analysis within CIFER® is the extraction of a complete multi-input/single-output (MISO) frequency-response database that employs an advanced multivariable spectral analysis with the Chirp-Z or "Zoom" transform (a computationally efficient Fast Fourier transform algorithm) and a composite optimal window technique. A flow-chart of the identification procedure is presented in Figure 3.

Unlike other frequency response software packages, CIFER® readily distinguishes primary controls/inputs from secondary inputs (i.e., correlated controls), producing MISO responses that are equivalent to the single-input/single-output (SISO) frequency responses that would have been obtained with only a single control input. Moreover, CIFER® takes the results from repeated conditioning of [the same] data using various window sizes and combines the auto- and cross-spectral density estimates using a non-linear, least-squares optimization to achieve a set of composite frequency responses.[5] These frequency responses can then be used to describe the dynamic characteristics of highly coupled aircraft without making *a priori* assumptions about a system's structure or order (i.e., a non-parametric model, typically represented as a transfer function). Selected frequency responses are typically presented in the form of a Bode plot (a plot of the input-to-output log-magnitude and linear-phase versus log-frequency) and a coherence plot. The coherence, or more appropriately the partial coherence for a MISO frequency response, may be physically interpreted as the fraction of the output power that is linearly related to the input power with the linear effects of the correlated input.[6] In general, the coherence is less than unity due to:

- low signal-to-noise ratio
- secondary/off-axis inputs
- remnants not accounted for by the 1st harmonic describing function.

Model Structure Determination

Model structure is determined in CIFER® using a nonlinear pattern search (Secant) algorithm to match

MISO frequency-responses (from either closed-loop or open-loop data). Identification of the model parameters is an iterative process, and involves minimizing the weighted square-error between the measured and the model's frequency responses. The frequency ranges for each input/output frequency response are selected individually based on an overall range of "good" coherence, and weighted based on the values of the coherence at each frequency point. Initial parameter values for the model may be obtained from one of three sources:

1) direct inversion of transfer-function model fits of the frequency-response data

2) equation-error identification results

3) *a priori* values based on simulation models or other data sources.[7]

A minimally parameterized model is achieved by identifying and removing parameters that have large insensitivities (i.e., parameters to which the converged solution is insensitive), or highly correlated to other parameters. A detailed description of the identification steps used for this model development may be found in Reference 8.

Time Domain Verification

The final step in the identification process (see Figure 3) was model verification. Verification was accomplished within CIFER® by driving the identified state-space model with flight data not used in the identification process; this is done to check the model's predictive capability for input forms other than the one used during the identification. CIFER® determines the unknown state equation biases and zero shifts and performs a non-iterative minimization of the weighted least square error between the model and vehicle responses.

Results and Conclusions

General

Time history plots of the [identified] minimally parameterized models for dissimilar flight-test data (for these cases, doublet inputs) are presented in Figures 4, 5 and 6. Additionally, the identified derivatives are listed in Table 2 along with wind tunnel results from Reference 1 for comparison. The conditions for the wind tunnel results are as follows:

- GW = 2750 lbs
- c.g. = 29.5% MAC
- airspeed = 104 KCAS
- H_d = 0 ft

The identified values were fit over a frequency range of 0.2 to 12 rad/sec; both the lateral/directional and longitudinal models were initialized with the derivative values from the wind tunnel study (Ref. 1). Derivatives corresponding to the throttle control axis were not identified due to the highly non-linear behavior of the engine control system. Similarly, the coupled 6 DOF derivatives were dropped from the identification since the corresponding frequency-response pairs showed no linear relationships.

Longitudinal Model

As shown in Figure 5, the results for the identified longitudinal model closely predict the aircraft's response in both magnitude and phase. The derivatives compare favorably with the wind tunnel values. Notable exceptions include the values for Z_u, M_u and M_w in which the identified values are all higher than the wind tunnel values. These identified values, along with $Z_{\delta e}$ (dropped due its large insensitivity), are considered to be questionable since there was insufficient variation in forward velocity (i.e., a low signal-to-noise ratio) to accurately determine the values from flight data.

Lateral/Directional Model

As shown in Figures 6 and 7, the results for the identified lateral/directional model closely predict the aircraft's "coupled" response in both magnitude and phase. The results for 15 out of the 17 identified lateral/directional derivatives compare favorably with the wind tunnel values. $N_{\delta a}$ and $Y_{\delta r}$ were the only two notable exceptions. The positive value for the identified $N_{\delta a}$ suggests that the aircraft exhibits a direct couple (or "proverse" yaw), which is contrary to the "adverse" yaw indicated by the wind tunnel results. The identified values for both $N_{\delta a}$ and $Y_{\delta r}$ may be explained by the fact that the sweep input is dynamic and of small amplitude, as opposed to a static, large amplitude input for the wind tunnel case.

Recommendations

1. Further frequency-domain testing should be conducted to resolve signal-to-noise issues related to the identification of Z_u, $Z_{\delta e}$, M_u and M_w.

2. Future efforts should be directed at developing a gain scheduling routine that would tie together derivatives identified at select airspeeds to form a [single] state-space model that is applicable over a wide airspeed range.

References

1. Teper, Gary L., "Aircraft Stability and Control Data," Contract NAS 2-4478, Systems Technology, Inc., April 1969 (Available as NASA CR-96008)

2. Tischler, Mark B., "Identification and Verification of Frequency-Domain Models for XV-15 Tilt-Rotor Aircraft Dynamics in Cruising Flight," AIAA Journal of Guidance, Control, and Dynamics, July 1986.

3. Bolding, Randy L., "Handling Qualities Evaluation of a Variable Stability Navion Airplane (N66UT) Using Frequency-Domain Test Techniques," M.S. thesis, The University of Tennessee Space Institute, December 1998.

4. ANON., "The Variable-Response Research Aircraft," Princeton University, no date.

5. Tischler, Mark B., "Frequency-Response Method for Rotorcraft System Identification: Flight Applications to BO-105 Coupled Rotor/Fuselage Dynamics," Journal of the American Helicopter Society, Vol 37, No 3, pgs 3-17, July 1992.

6. Conference Publication, NASA CP 10149, USAATCOM TR-94-A-017, "Comprehensive Identification from Frequency Responses," Class Notes from short course held at NASA Ames Research Center, Moffett Field, California, September 19-23, 1994.

7. Fletcher, Jay W., Wolfgang von Grunhagen, Jurgen Kaletka, and Mark B Tischler, "Time and Frequency-Domain Identification and Verification of BO 105 Dynamic Models," 15th European Rotorcraft Forum, Amsterdam, September 1989.

8. Catterall, R. Charles, "Development of a State-Space Model for the Ryan Navion (N66UT) Using Frequency-Domain Techniques," M.S. thesis, The University of Tennessee Space Institute, August 1999.

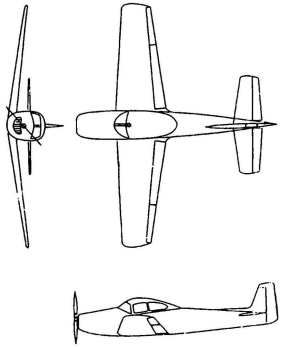


Figure 1: Three view drawing of the Navion

Table 1: Data collected for aircraft identification analysis

Inputs:

- aircraft control surface deflections
  - aileron deflection (δ_a)
  - elevator deflection (δ_e)
  - rudder deflection (δ_r)
- throttle control deflection (δ_T)

Outputs:

- body attitudes
  - roll attitude (ϕ)
  - pitch attitude (θ)
- body angular rates
  - roll rate (p)
  - pitch rate (q)
  - yaw rate (r)
- linear accelerations
  - longitudinal acceleration (a_x)
  - lateral acceleration (a_y)
  - normal acceleration (a_z)
- speed data (reconstructed from kinematic relationships)
  - longitudinal speed component (u)
  - lateral speed component (v)
  - normal speed component (w)

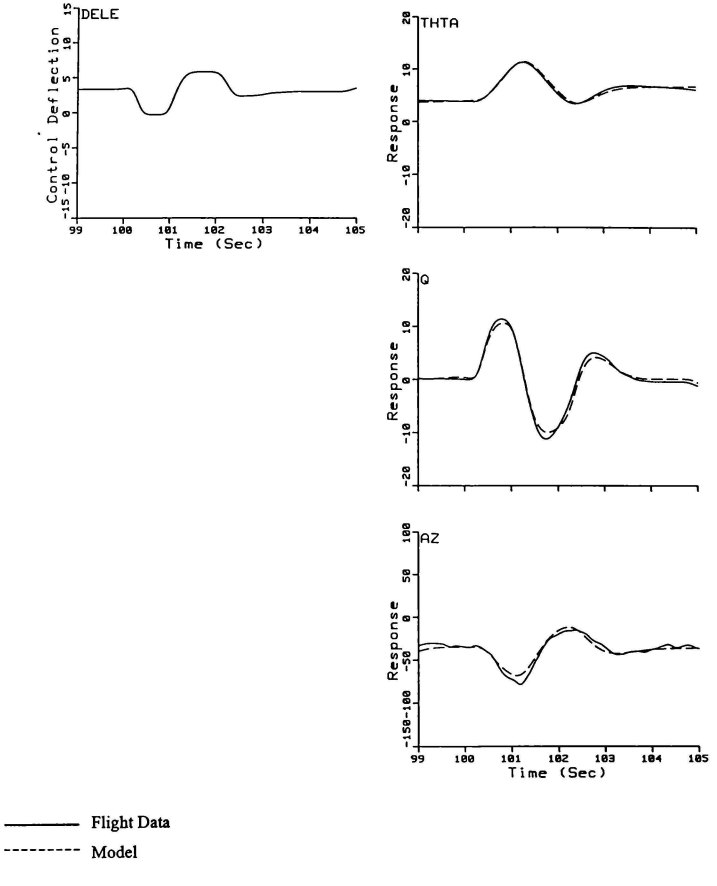


Figure 4: Comparison of longitudinal response time histories between flight data and model for a pitch doublet

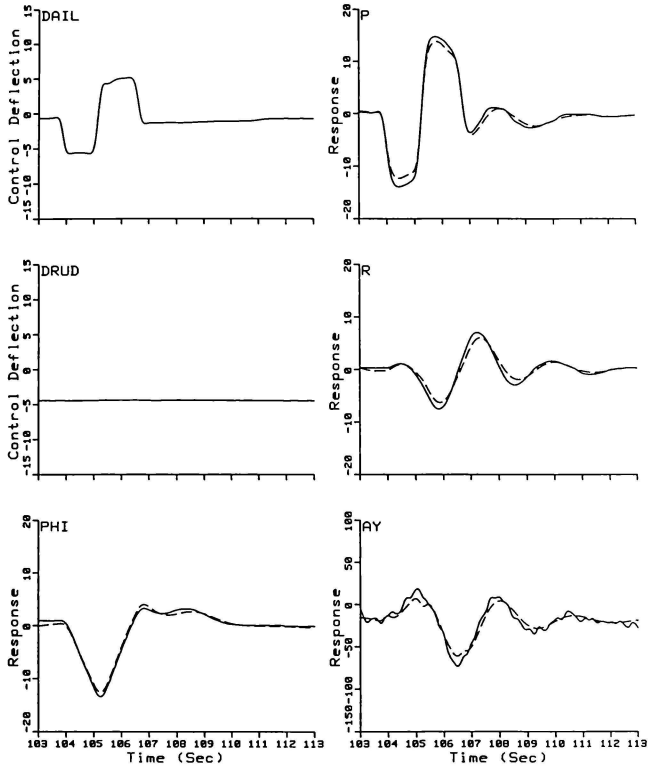


Figure 5: Comparison of lateral/directional response time histories between flight data and model for a roll doublet

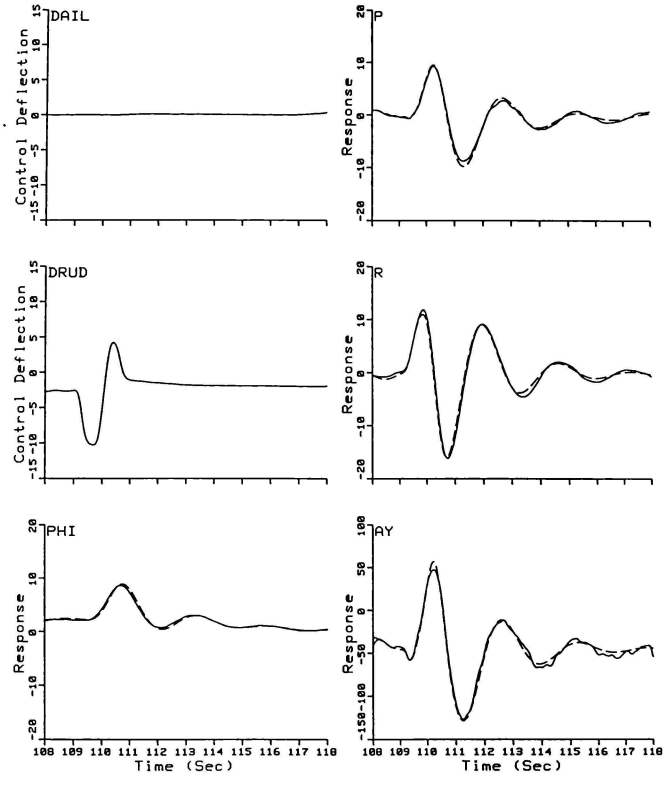


Figure 6: Comparison of lateral/directional response time histories between flight data and model for a yaw doublet

Table 2: Comparison of identified and wind tunnel stability and control derivatives

	Derivative		Frequency-domain parameter values from present study	Wind tunnel parameter values from Reference #
	Name	Unit		
Longitudinal Derivatives	X_u	1/sec	-0.0972	-0.0451
	X_w	1/sec	0.0000 †	0.0361
	Z_u	1/sec	10.37	-0.3697
	Z_w	1/sec	-3.2840	-2.0240
	Z_q	1/sec	0.0000 †	0.0000 *
	M_u	1/(sec · ft)	0.1568	0.0018
	M_w	1/(sec · ft)	-0.4648	-0.0390
	M_q	1/sec	-2.9250	-2.8610
	$X_{\delta e}$	ft/(sec^2 · rad)	0.0000	0.0000
	$Z_{\delta e}$	ft/(sec^2 · rad)	0.0000 †	-28.17
	$M_{\delta e}$	1/sec^2	-14.20	-11.19
	$\tau_{\delta e}$	sec	0.0000 †	0.0000 *
Lateral/Directional Derivatives	Y_v	1/sec	-0.0931	-0.2540
	Y_p	1/sec	0.0000 †	0.0000 *
	Y_r	1/sec	1.6170	0.0000 *
	L_v	1/sec	-0.0593	-0.0910
	L_p	1/sec	-5.8160	-8.4020
	L_r	1/sec	1.8540	2.1930
	N_v	1/sec	0.0299	0.0250
	N_p	1/sec	-0.5475	-0.3498
	N_r	1/sec	-0.9526	-0.7605
	$Y_{\delta a}$	1/sec	0.0000 *	0.0000 *
	$Y_{\delta r}$	1/sec	3.4690	12.45
	$L_{\delta a}$	1/sec^2	15.74	28.98
	$L_{\delta r}$	1/sec^2	1.3800	2.5480
	$N_{\delta a}$	1/sec^2	0.4862	-0.2218
	$N_{\delta r}$	1/sec^2	-4.2880	-4.5970
	$\tau_{\delta a}$	sec	0.0000 †	0.0000 *
	$\tau_{\delta r}$	sec	0.0000 †	0.0000 *

* Eliminated from model structure based on *a prior* assumptions

† Eliminated during model structure determination base on high insensitivity or high correlation

DESIGN AND FLIGHT TESTS OF A GROUND COLLISION AVOIDANCE SYSTEM

G. Sachs* and U. Sennes†, Technische Universität München, Munich, Germany
H.-L. Dudek‡, Euro Telematik GmbH, Ulm, Germany

ABSTRACT

A ground collision avoidance system is described which is based on digital terrain modeling and flightpath prediction. Flightpath prediction yields possible collisions with the terrain and obstacles well in advance so that an early warning is possible. Further, the system provides the pilot with information about evasive maneuvers (climb or turn).

An experimental evaluation was performed which included flight tests in a demanding terrain environment near Innsbruck/Austria in the Alps. The test vehicle was the research aircraft ATTAS of the German Aerospace Center DLR. The results of the experimental evaluation show that all design goals were achieved.

INTRODUCTION

Controlled Flight Into Terrain (CFIT) is still an important category of accidents. An accident is termed a CFIT when an otherwise serviceable aircraft is under control of the crew, but is flown unintentionally into terrain, obstacles or water. Because of misorientation and/or poor visibility the crew has no prior awareness of the impending collision (Ref. 1).

To reduce the number of CFIT accidents Ground Proximity Warning Systems (GPWS) were developed. Since their introduction about 20 years ago the accident rate has considerably gone back, Fig. 1. However, there is still a significant number of CFIT accidents, Fig. 2.

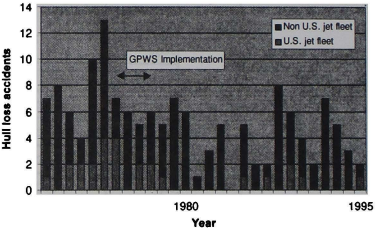


Fig. 1: Percentages of Controlled Flight Into Terrain (CFIT) and other fatal accidents, from Ref. 2

This situation can be related to the function of current GPWS which is based mainly on measurements of a radar altimeter looking downwards. With reference to the distance between terrain and aircraft and the change in distance, the system determines conditions indicating a hazard. If specified limits are exceeded warnings are given. However, this type of GPWS has problems with regard to late and unwanted warnings (Ref. 3). No terrain elevation can be dedected ahead of the aircraft. Especially in steep mountainous areas the precaution time may be very short.

* Prof. Dr.-Ing. Gottfried Sachs, Director, Institute of Flight Mechanics and Flight Control. Fellow AIAA.

† Dipl.-Ing. Ulrich Sennes, Research Assistant, Institute of Flight Mechanics and Flight Control.

‡ Dr.-Ing. Heinz-Leo Dudek, Managing Director, Euro Telematik GmbH.

Copyright © 1999 by G. Sachs, U. Sennes and H.-L. Dudek. Published by the American Institute of Aeronautics and Astronautics Inc. All rights reserved.

To enhance the capability of such systems, a research and development program was conducted which was financially supported by the Federal Government of Gennany. The partners of this program are Daimler-Benz Aerospace AG, Deutsches Zentrum für Luft- und Raumfahrt, Elektroniksystem- und Logistik-GmbH and Technische Universität München.

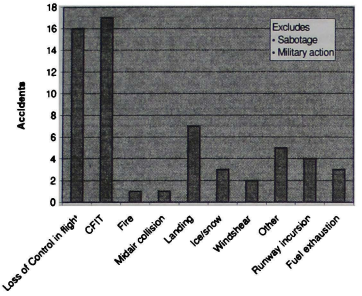


Fig. 2: CFIT hull loss accidents worldwide from 1987-1996, from Ref. 2

SYSTEM DESCRIPTION

GCAS Concept

A concept for a Ground Collision Avoidance System (GCAS) was developed which uses digital terrain elevation data and satellite navigation. Due to flight path prediction a possible collision with the terrain can be detected early so that warnings can be given in time. Aircraft performance and actual flight conditions are taken into account. Because of the terrain database the elevation not only ahead of the aircraft but also to the side is known. So it is possible to provide the pilots with information about extended possibilities of evasive maneuvers, e.g. turning instead of climbing to avoid the collision, Fig. 3.

There are several advantages of this new concept:

- early warnings, Fig. 4A
- avoidance of unwanted warnings, Fig 4B
- extended possibilities of evasive maneuvers, Fig. 3
- improvement of situational awareness

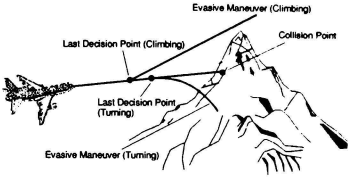
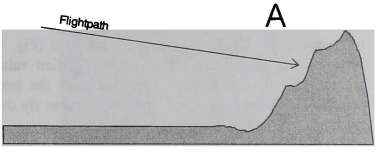


Fig. 3: GCAS concept



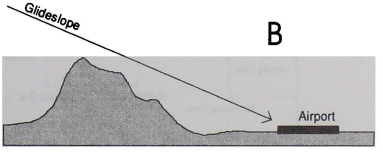


Fig. 4: Possibilities of GCAS concept
A Early warnings in mountainous areas
B Avoidance of unwanted warnings

Fig. 5 shows a schematic structure of the GCAS concept. It uses a terrain database, aircraft performance data (e.g. for climbing) and flight vector data to determine possible collisions and to provide warnings and indications for evasive maneuvers.

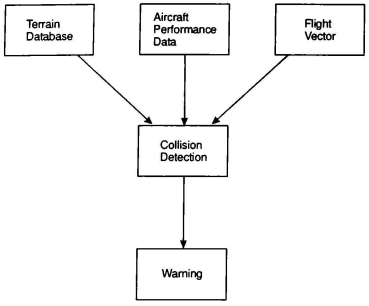


Fig. 5: Schematic structure of GCAS concept

Various warning/hazard degrees are specified with reference to the time of a potential collision (Fig. 6). Prewarning and warning times set at specified values are introduced. The evasion time indicates the latest possibility to start an evasive maneuver and to fly over the obstacle in a safe distance.

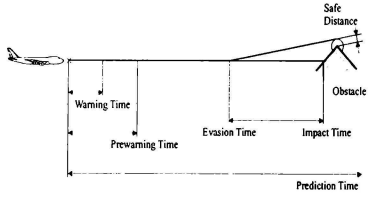


Fig. 6: Time sections

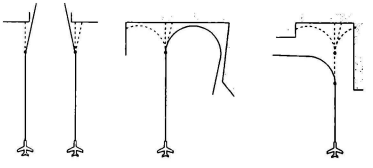


Fig 7: Lateral evasive maneuvers

If an evasive climb is not possible, the terrain on the sides of the aircraft are scanned for turning possibilities. There may be various forms of turning evasive maneuvers, as illustrated in Fig. 7. Changes with less heading are preferred.

Data Management

An important part of the GCAS concept concerns the management of the digital terrain database. The database stored on board of the aircraft covers at least the area which is intended to be flown over, but is designed for worldwide coverage. To reduce the amount of data, a compression technique was applied.

Data decompression takes place on board of the aircraft. Therefore, it is necessary to apply a technique which requires little time for performing this task in real-time. Another feature of the data management technique concerns different accuracy levels for describing the terrain. In areas of flight close to the ground like airports, the accuracy can be set at a higher value than in other areas.

The data management of the GCAS concept is referenced to Ref. 4 which specifies standards for data processing. Accordingly, the database quality has to fulfil requirements in regard to

- accuracy
- resolution
- availability
- confidence
- actuality (relevance to the present)
- validation
- verification

Display Configuration

GCAS warning and guidance information is presented to the pilot on a display. Two display configurations were considered:

I segment configuration
II bitmap configuration

Display configuration I shows a semicircular arrangement of segments which serve as an indication of areas in front of the aircraft (Fig. 8). A possible hazard is indicated by marking a segment with a colour (red and orange). The two ring arrangement provides the pilot with an information about the distance (30 sec flight time for the inner ring, 120 sec for the outer ring). An appropriate evasive maneuver is indicated at the top.

Display configuration II provides more detailed information about the terrain and evasive maneuvers by showing a bitmap presentation (Fig. 9). The display provides the pilot with information about his position, the flight direction of the aircraft and dangerous areas in a manner which yields a quantitatively precise knowledge. Dangerous areas are indicated by colours which vary according to the degree of danger. The actual position is marked by an aircraft symbol, and semicircles are shown to support the pilot in estimating distances.

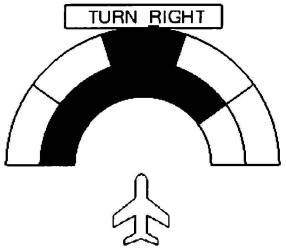


Fig. 8: Segment display configuration

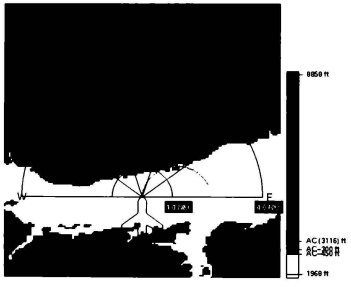


Fig. 9: Bitmap display configuration

EXPERIMENTAL EVALUATION

For evaluating the GCAS concept an airworthy prototype which is shown in Fig. 10 was developed and built according to RTCA DO-160D (Ref. 5). It was tested in flight tests with the twin jet engines research aircraft ATTAS (Advanced Technologies Testing Aircraft System) of the German Aerospace Center DLR, Fig. 11. The GCAS prototype installed in the aircraft is shown in Fig. 12.

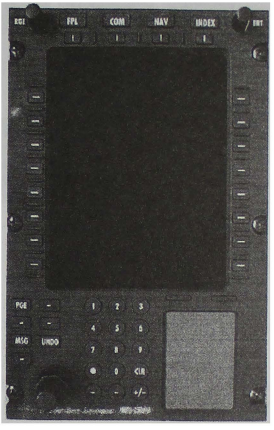


Fig. 10: GCAS prototype



Fig. 11: ATTAS Research aircraft.

Simulation experiments at the flight simulator of the Institute of Flight Mechanics and Flight Control of the Technische Universität München were conducted prior to the flight tests. They showed that the system worked properly. Eight pilots took part in the simulation experiments.

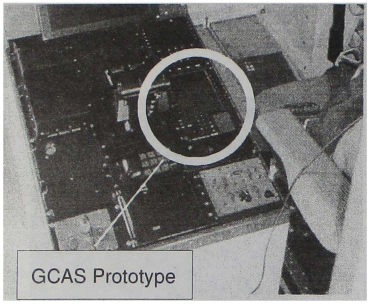


Fig. 12: GCAS prototype installed in aircraft

The flight tests took place at Innsbruck airport area in March 1998. Fig. 13 shows the test area in the Inn river valley which extends from west to east. This area was chosen because of its topography yielding a demanding test environment for the evaluation purpose in mind.

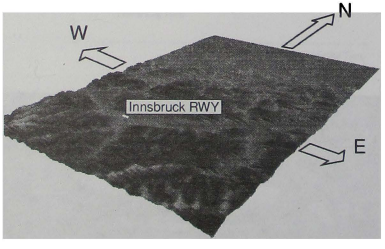


Fig. 13: Innsbruck test area

A series of test flights were performed with different experimental scenarios. As an example, one of these is described in the following. The trajectory is shown in Fig. 14, indicating nine sections:

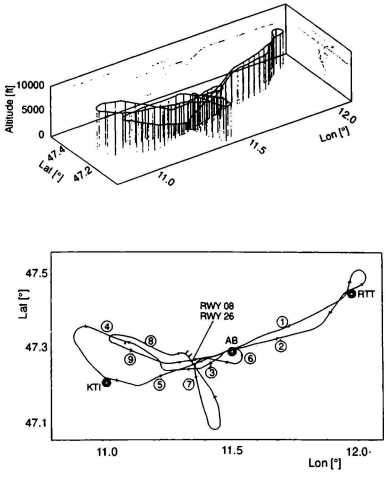


Fig. 14: Trajectory of test flight

1 Departure from RWY 08 to RTT.
2 Approach to RWY 26, descending intentionally too late and too steep, go around at decision height over AB.
3 Circling approach to RWY 08 under VMC with an extended downwind leg (i.e. a flight towards a mountain)
4 Flight in the Inn river valley under VMC to KTI.
5 Approach from KTI to RWY 26.
6 Circling approach to RWY 26 under VMC, break off before aerodrome, turn to the south.
7 Flight into the southern valley, turn back.
8 Flight back to the north towards a mountain.
9 Approach and Landing on RWY 08.

The GCAS system showed the following reactions which were evoked due flight maneuvers intentionally performed:

At the first approach (section 2) a "Terrain" prewarning was provoked by an intentionally fast descent. Additionally, the advisories "No Turn left" and "No Turn Right" were indicated.

At the circling approach (section 3) which included an extended downwind leg the warnings "High Terrain Ahead" and "Turn Sideways" appeared.

The circling approach of sections 6 in the narrow Inn river valley was conducted at a speed which was intentionally too high. As a consequence, the system generated a "High Terrain Ahead" prewarning followed by a "Turn Sideways" warning.

Another system test concerns section 7. When flying into the southern valley a "Terrain" prewarning was generated. After 16 seconds a "Pull Up" warning appeared. They were evoked by an intended flight towards a terrain elevation. This condition corresponds to a typical CFIT situation.
Similarly, the warnings "High Terrain Ahead" and "Turn Sideways" were indicated at the flight back in northern direction towards a mountain (section 8).

In summary, all prewarnings, warnings and advisories were correct. The system worked in accordance with the specifications.

CONCLUSIONS

An experimental ground collision avoidance system (GCAS) is described which is based on digital terrain modelling and on flightpath prediction. It can identify potential hazard and danger situations by comparing the predictive flightpath with a stored terrain database. The developed flightpath prediction technique which takes the actual flight condition and the available performance of the aircraft into account enables the system to identify a possible collision with the terrain or obstacles well in advance so that an early warning is possible. Because of the knowledge of the surrounding terrain (including obstacles) it is further possible to provide the pilot with information about appropiate evasive maneuvers, e.g. a climb or a turn.

The experimental evaluation of the system included simulation and flight tests. The flight tests were performed with the ATTAS research aircraft of the DLR German Aerospace Center. The test area was located near Innsbruck/Austria in the Alps, representing a demanding test environment because of the surrounding mountains. The results of the experimental evaluation yield a confirmation of the developed concept for a ground collision avoidance system.

REFERENCES

1 Duquette, A.: Safer Skies. FAA News, April, 1998.

2 Carbaugh, D. - Cooper, S.: Avoiding Controlled Flight Into Terrain. Airliner – The Boeing Company, pp. 1-11, April-June, 1996.

3 Bateman, D.: Development of Ground Proximity Warning systems (GPWS). Royal Aeronautical Society, Control Flight Into Terrain – One day conference, November 8, London,. 1994.

4 RTCA DO-200A Standards for Processing Aeronatical Data. Radio Technical Commission for Aeronautics, September, Washington, 1998.

5 RTCA DO-160D Environmental Conditions and Test Procedures for Airborne Equipment. Radio Technical Commission for Aeronautics, July, Washington, 1997

USE OF THE COCKPIT-TRANSFORMED SIMULATOR FOR TESTING OF PROPOSED ALGORITHMS OF PILOT'S CONTROL ACTIONS DURING AIRCRAFT MANEUVERING SUPPORT SYSTEM

Oleg A. Yakimenko*, Vladimir V. Demidov†
Aviation and Aeronautics Academy of Sciences, Air Force Engineering Academy, Moscow, Russia
Naval Postgraduate School, Code AA/YK, 699 Dyer Rd., Monterey, CA 93943-5106
phone (831)-656-2826; fax (831)-656-2313; e-mail oayakime@nps.navy.mil

Paper continues the previous researches, which deal with the design of specialized mobile adaptive training/simulation system (SpMATS) for modern multi-regime aircraft and its implementation for pilot's control actions during maneuvering support system (PSS) algorithms testing. Paper reminds the main ideas, general structure and some features of SpMATS, and also discloses its possibilities for different algorithms proposed for future on-board utilization research. The main part concerns the results of implementation, verification and semi-natural testing of PSS algorithms on different stages of flight on board of combat aircraft within the frame of SpMATS utilization, and evaluates PSS's helpfulness and effectiveness.

Introduction

It's well known that the active search of new approaches and methods of flight simulation and pilots' preparation is the distinctive feature of modern stage of aircraft (a/c) utilization. However obtaining of appreciable results is impossible without the new approaches and consecutive introduction of science and engineering achievements in the process of practical decision of the delivered problem. One of the most perspective means for such purposes it's considered the built-in into on-board equipment and integrated with it means providing simulations and trainings on airfield parking called embedded training.

But the built-in simulation/training equipment is an aircraft subsystem (hardware and software) not necessarily integrated into onboard equipment that exists as additional mode of operation. E.g. SpMATS[1] was created as a specialized cockpit transformed simulator of "built-in" type.

Initially it was intended for simulation and training of a complex of problems of multi-regime a/c application in a cabin of real a/c, which is under a current but without engines switching on[2].

But besides flight control and weapon application skills training SpMATS's adaptive object-oriented software can provide a semi-natural research of new aerodynamic schemes, control systems and others algorithms[3], including PSS's.

PSS is considered as a subsystem of on-board universal pilot's support system (electronic co-pilot, pilot's associate), which provides pilot's control actions support during performing of more or less long-term spatial maneuvers, such as take-off and climbing, flight on a route in condition of prohibited zones of flight presence, surface target attack, air target guidance, descent and landing.

So SpMATS was implemented for PSS's algorithms[4] semi-natural testing.

Thus the present paper deals with the use of the cockpit-transformed SpMATS for the testing of proposed PSS's algorithms and is organized as follows. The first section contains a brief description of SpMATS and its software filling, the second part gives a general characteristic of semi-natural experiment, the third section describes the results of semi-natural experiments.

I. Hardware and software description

I.1. SpMATS structure

SpMATS represents a complex from the multi-regime a/c, computing system (CS) and pilot visualization system (PVS) (Fig.1). CS consists of two field PC. First PC carries out a modeling process of a/c movement, synthesis of cockpit panel devices' images, uniform indication system (UIS) images, ground and air targets and outcabin virtual space. The source information for modeling is the data on controls' position received through a/c's regular information systems. The a/c controls data inputs to this PC through the two information cables plugged in the same board of data input installed in standard ISA trunk slots. A training instructor/researcher uses the second PC for the management of simulation/training process. Both PC are connected to the standard bi-directional port RS-232 with the third cable (Fig.2), and use the specially developed protocol.

PVS represents a flat color monitor, connected via standard cable to modeling PC and installed with the help of an arm (rubber bandage) in front of the UIS block in a/c cockpit (Fig.3).

For normal SpMATS functioning it's enough to link the a/c under current, to switch on the objective control system (OCS), and to unbreak the control column

---

* Doctor of Technical Sciences, Professor, AAAS Corresponding Member. Currently on National Research Council Associateship program tenure in Naval Postgraduate School.
† PhD graduate.

Copyright © 1999 The American Institute of Aeronautics and Astronautics Inc. All rights reserved.

(stick) with observance of all security measures agreed to the Flight Operation Manual.

The pilot, sitting inside the a/c cabin, observes on the color monitor an outside virtual space, targets, basic devices and indicators of an instrument board, and also all graphic UIS images, appropriate to a mode of flight and corresponded to simulation/training task. He operates with the a/c and armament control system (ACS) models by usual controls without start of engines.

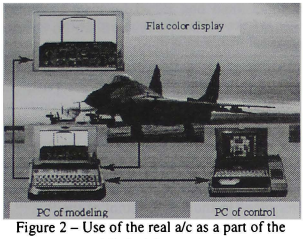


Figure 2 – Use of the real a/c as a part of the mobile training system

The instructor carries out a choice of flight task and a control of training process (experiment). Instructor's PC analyses the process of simulation/training, fixes modeling data (pilot's infringements and mistakes), creating on their basis a corresponding databases for further analysis.

I.2. PSS software

SpMATS software in general includes: the algorithms and programs describing complete 6-DOF model of a/c movement on a runway and in the air (the object-oriented approach allows to change an a/c type rather quickly); the models of its subsystem (locator, head-up display, armament system, jet power system, etc.); the model of standard atmosphere and different wind disturbances; the algorithms and programs of outside cabin virtual world formation and visualization, the programs of live cockpit panel devices; and special algorithms of PSS.

This software in the whole allows to simulate dynamics of a/c spatial movement in real time scale with the high degree of affinity to real objects both simulated a/c dynamics and virtual reality of inside and outside cabin environment.

Mentioned PSS's algorithms[5] provide short-cut time scale solution of different optimizational tasks with the help of several "fast" modifications of the direct method of variational calculus[6-8]. This software is designed for such more or less long-term maneuvers as: take-off and climbing, flight on a route, surface-based target attack approach, guidance on moveable object, air combat, descent and landing approach.

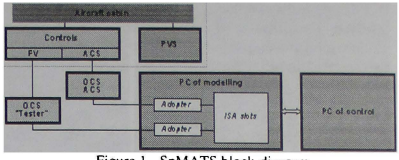
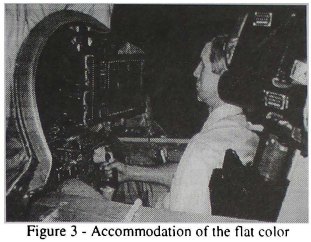


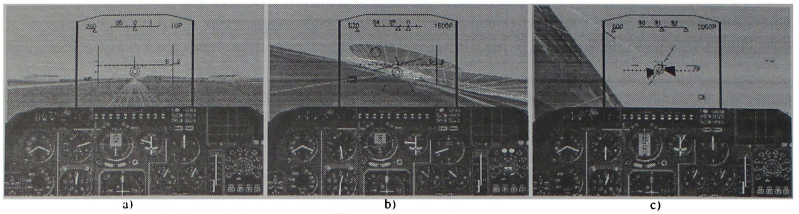
Figure 1 - SpMATS block-diagram


Figure 3 - Accommodation of the flat color monitor in front of UIS block

It's supposed that all this particular tasks (except flight on a route) have been preliminary formalized in order to form the grid of calculate nodes; then in this nodes variational trajectory task have been solved by means of these modifications of direct method at powerful stationary computers; obtained during solution defining reference near-optimal (NO) trajectories parameters (only a few for each trajectory!) have been stored in on-board computer in the image of "bank of zero guess to optimal trajectories" (in case when the goal function is not unique there would be several trajectories banks).

In real flight (during simulation with the help of SpMATS) for real strategic and/or tactic conditions for real aircraft, its subsystems and even pilot state a concrete bank is chosen and following multi-dimensional interpolation is performed. Then with the help of determined from "trajectories bank" zero approximation concrete optimizational problem is solved in accelerated scale of time (it takes no more than 2% from trajectory time itself for Pentium-100 processor). Reconstructed by this way reference trajectory is presented to a pilot in cognitive image of road-in-the-sky (RiS) on display for its following tracking[4,5] (Fig.4). RiS is equipped with appropriate wayside signs like "Throttle-up", "Throttle-down", "Fire", "Drag flaps", "Gears", etc.

For the flight on a route in case of prohibited zones of flight presence there is used another modification of the direct method based on route, profile and velocity histories' spline approximation with subsequent variation and use of global extremum search method[8].



a) b) c)

Figure 4 - Examples of piloting at different stages of flight with RiS assistance (take-off (a), flight on a route (b), surface-based target attack approach (c))

II. General characteristic of semi-natural experiment

In total 14 pilots took place in the semi-natural experiment on PSS' algorithms testing*. Ten of them were the pilots of strike a/c and fighters, and four were the pilots of transport aviation (five had 3rd class classification, six - 2nd class, three - 1rd class). The average flight hours were 300-1500*hrs*.

During experiments the pilots were asked to track different recommended by PSS reference trajectories (RT) with the use of RiS prompt. 135 realizations have totally been received (see Table 1).

Table 1.

Task	Amount of realizations
takeoff/climbing	*5*
flight on a route	*20*
surface-based target attack approach	*30*
air target guidance	*20*
landing approach	*20*
glissade	*20*
test trajectories	*20*

During experiments there were optimized some PSS parameters, namely: RiS's cross section's size, RiS type (gutter, tunnel); image type (real, compressed). Of course all experiments include real atmosphere with different kinds of disturbances (constant wind, wind shift, turbulence) imitation.

For each stage of flight (type of trajectory) there were analyzed the following parameters: accuracy of trajectory observation, accuracy of terminal conditions satisfaction, pilot's control actions character, dependence on flight regime, etc.

The accuracy of the RT tracking was estimated by the mean integral distance from the RT (RiS's center line) I_D and average integral deviations of the trajectory's parameters from the required values (I_a and $I_{|a|}$):

$$I_D = \frac{1}{T}\int_T \sqrt{\sum_{i=2}^{3}\left(x_i(t) - x_i^{ref}(t)\right)^2}\,dt\,,$$

$$I_a = \frac{1}{T}\int_T \left(a(t) - a^{ref}(t)\right)dt\,, \qquad (1)$$

$$I_{|a|} = \frac{1}{T}\int_T \left|a(t) - a^{ref}(t)\right|dt\,.$$

Here $a(t)$ is the current value of any parameter, while its reference value $a^{ref}(t)$ and reference states' values are determined in the nearby point of the RT.

III. Semi-natural experiment results

III.1 Vigorous trajectories tracking

The most difficult for tracking, as it was expected, there were vigorous trajectories, namely the trajectories of surface-based target attack approach.

Semi-natural modeling of the surface-based target attack approach was carried out as follows. The recommended trajectory was settled from the set of relative entry conditions. At the end of attack approach maneuver and at exit into the required final conditions the pilot, seeing the target on PVS, with the help of simulated ACS indication was being carried out the targeting and application of the weapon, as well as subsequent exit from the attack.

On Fig.5 there is submitted the example of recommended trajectory and results of its ten realizations by the pilots for the case of attack of the single motionless target by unguided rockets (on this and others figures $0x_1x_3$ is the local horizon plane, $0x_1x_2$ is the local vertical plane, V is the airspeed, $\bar{n}$ is the engine's rotor relative revolutions, n_{ya} is the load factor, γ_a is the bank angle). As a final parameters of the trajectory there were chosen: airspeed 800*km/h*, range to the target 3600*m*. The recommended trajectory was tracked by the pilot with use of RiS, which cross section was smoothly changed from 100x50*m* at the initial point up to 50x25*m* in the final point of trajectory, i.e. exit on the targeting line (TL).

The qualitative comparison of parameters of the recommended trajectory and its realizations shows that they are enough close to each other. Averaged for all received realizations results of the numerical analysis of parameters' deviation show the following integrated accuracy of the RT tracking:

* Some estimates and recommendations were also given by test-pilots Heroes of Russia V.Pugatchev, A.Bichkov, A.Kutuzov, I.Votintzev, I.Solovjov, and O.Tzoj.

$$
\begin{aligned}
&I_D = 62 \pm 20m, && I_{|\Delta n_y|} = 0.65 \pm 0.10, \\
&I_{\Delta V} = -11.7 \pm 3.9 m/s, && I_{\Delta \gamma} = 1.2 \pm 3.9^{\circ}, \\
&I_{\Delta n_y} = 0.049 \pm 0.177, && I_{|\Delta \gamma|} = 13.20 \pm 0.65^{\circ}, \\
&\Delta T = 3.0 \pm 2.6s.
\end{aligned} \qquad (2)
$$

Final TL-angles discrepancies both in vertical and horizontal planes -

$$\Delta\varepsilon_f^{\gamma} = -2.3 \pm 1.0^{\circ} \text{ and } \Delta\varepsilon_f^{h} = -2.1 \pm 0.94^{\circ} \qquad (3)$$

- were small enough to be eliminated by the pilot in the process of following targeting.

The values of the load factor and bank angle along the trajectory on the average practically coincide with the required values and oscillate slightly near them. That is due to some "redistribution" of controls the pilots easily parry the disturbances brought in by them or turbulent atmosphere in order to track the RT accurately.

Even the presence of mistake in the moment of thrust switching (Fig.5), influencing a little bit on airspeed vs. time history, did not complicate the RT tracking. The result of this mismatch was either insignificant increase of maneuver time (in average at $\Delta T = 3.0 \pm 2.6s$ when the reference time of maneuver was $45s$), or on the contrary the reduction of the maneuver time on 5...6 seconds with default of the final speed value.

On Fig.6 a/c deviations from the RT in longitudinal (normal) and lateral directions (i.e. in the direction of correspondent RiS's walls) are shown. From the comparison of these data with the stick deflection in longitudinal and lateral channels (Fig.7) during trajectory tracking it's seen, that the pilot aspires to not allow an exit for RiS's limits, interfering in control only at approach to RiS's walls and providing thus the movement along the RT. The correlation analysis of deviations of the trajectory parameters from the reference values shows the presence of steady interrelation between the state coordinates – mismatches with the RT – and controls deviations. The corresponding correlation factors are:

$$K_{\Delta n_y, \Delta y_r} = -0.5 \ldots -0.9\,, \; K_{\Delta y_r, \Delta z_r} = -0.5 \ldots -0.9\,, \qquad (4)$$

where the deviations from the RT Δy_r and Δz_r are defined in the direction of RiS walls. That is RiS's cross section dimensions play a very the important role.

It is necessary to note also, that the pilot observes the trajectory itself, not the reference values of the load factor and bank angle as it take place at usual director control mode. The information on bank angle mismatch is carried out by RiS's cross section orientation. The longitudinal control is carried out on the information about a/c position with respect to the top and bottom RiS wall. When the pilot sees the whole forthcoming trajectory, he can prepare for the appropriate control intervention beforehand. Besides in accordance to pilots' responses, the RiS's cross sections are perceived by them as some kind of the leading a/c, behind which they should fly. This circumstance allows them easily piloting the a/c even near RiS (behind its limits).

On pilots' responses, the additional psychophysiological load at RT tracking is insignificant. It also is confirmed by the results of controls' deviations and their spectral structure analysis (Fig.7). It is seen, that the control actions with the proposed director with foreseeing regime have a clear correcting character, providing a/c movement within the RiS's corridor. Thus the basic frequencies of the stick deviations are concentrated in the area up to $0.7Hz$, i.e. do not exceed the values for usual flight.

The results of performed realizations as well as pilots' interviewing show that the tracking of the NO RT with the help of RiS considerably facilitates the maneuver of attack approach performing and provides very accurate satisfaction of armament use conditions at the end of the trajectory. Having such support, the pilot is practically completely released from the necessity of difficult spatial maneuver construction by himself. And it's especially important in conditions of insufficient visibility, and moreover in conditions of invisibility (night, heavy rain, fog, etc.). So for example, the performance of the similar maneuver in conditions of the stand, at the lateral view absence (virtual reality is simulated within the frame of small monitor only) without the RiS prompt, represents the large complexity for the pilots (see corresponding trajectory on Fig.5). It is seen that the absence of visual contact with the target in first half of maneuver strongly complicates the a/c piloting. The pilot makes plenty of superfluous control actions. Especially it is appreciable in the lateral channel.

At observation of the recommended trajectories at enough rigid (small) dimensions of RiS's cross section there were some inadvertent exits for its limits (see example on Fig.6). At small deviations, when the road remained in a field of sight and continued to carry the information about a/c position with respect to the RT, the pilots either without difficulties came back into the RiS, or continued the movement near it up to the moment of an exit on the target. At the large deviations, when the road appeared outside of the screen, known complexities naturally occurred.

The exits for the RiS's limits occurred, mainly, due to small height of RiS's cross section, because of what the pilot had not enough time to dose correctly the amount of control actions before the a/c left the RiS. The researches have shown that the opportunity of trajectory tracking without an exit for its limits is determined by the presence at each instant of time the sufficient time reserve for intervention into the a/c control and elimination of a mismatch up to an exit for the RiS limits. The size of this reserve can be estimated by the time before crossing of any of RiS's wall under the constant controls, that is at constant, equal to current values of

the bank angle, normal and tangential projections of the sum load factor. The size of this time depends on RT parameters, RiS's dimensions, and also size of a mismatch of the current movement parameters from the required ones[4,9]. The time before crossing a RiS's wall at each moment at an exact tracking of the RT, i.e. at current mismatch absence, dependent only from its parameters and RiS's dimensions, can characterize an opportunity of trajectory observation without an exit for RiS's limits.

Fig.8 shows the dependencies of time before crossing the RiS's' wall $\Delta\tau_c$ both for the exact movement along the RT and for one of practical realizations. Bottom part of Fig.8 defines the critical wall, which could be "pierced" at the fixed control.

It is seen, that even at exact movement along the RT there are some points, in which $\Delta\tau_c$ reaches its minimal value of 3s (for the particular given trajectory and RiS's dimensions). These points are coincide with the points of maximal curvature of the RT, that is the points with marginal load factor and bank angle time-derivatives. Obviously, it raises the requirements to the corresponding restrictions on controls and their derivatives at NO trajectories calculation[5].

The critical time $\Delta\tau_c$ at real movement is much less, than at exact RT tracking because of current nonzero mismatch presence (except for several special pieces with a favorable combination of appropriate coordinates and controls of opposite deviations). It sharply changes at the moments of critical (dangerous) wall switching.

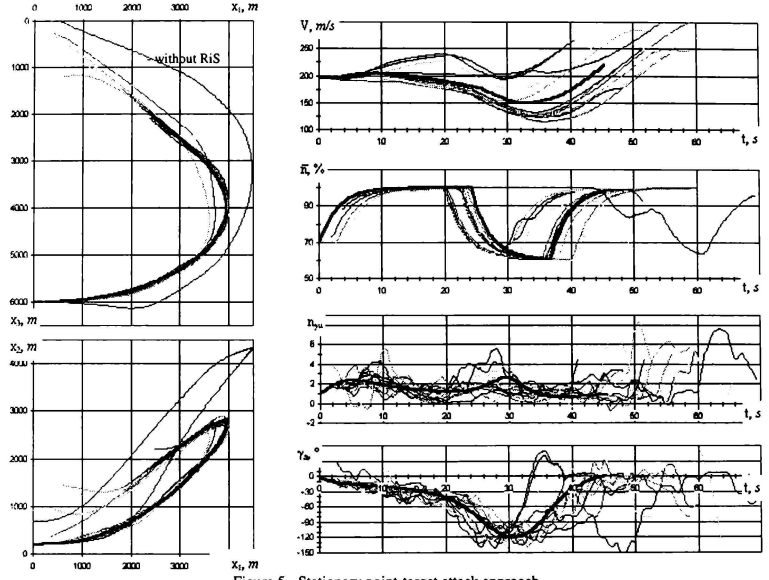


Figure 5 - Stationary point-target attack approach

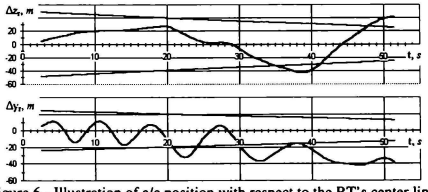


Figure 6 - Illustration of a/c position with respect to the RT's center line

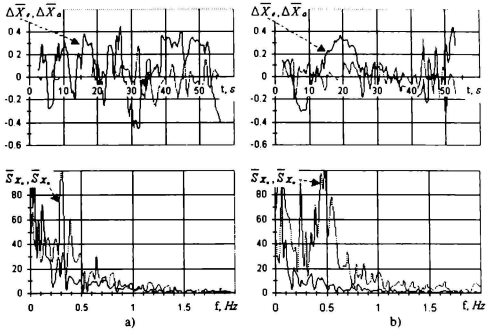


Figure 7 - Relative controls deflections and their spectral densities with RiS prompt (a) and without it (b)

The experiments have confirmed that the pilots master the new control mode with the RiS assistance fairly quickly. Usually for accusation of correspondent skills it's enough to have only a few flights on different trajectories. Fig.9 demonstrates the results of improvement of tracking accuracy for one of the pilots for several training days as well as within each day. These results confirm a high level of pilot's master capacity and skills accusation at piloting with the help of RiS. From this diagram it is seen, that already after a few trainings the value of mean-integral deviations from the RT aspires to value not exceeding 20...30*m*, i.e. the realized trajectory even for vigorous maneuvers in the average guaranteed passes within the RiS's limits.

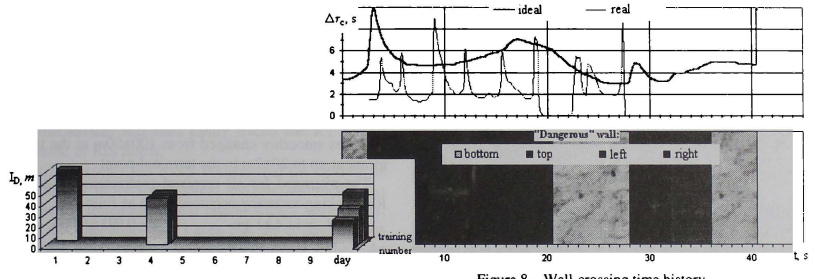


Figure 9 – Illustration of pilot's learning capacity

Figure 8 – Wall-crossing time history

Fig.10 represents the results of the other RT, namely runway attack approach tracking. It's seen that, having accustomed with a new kind of director prompt, the pilots traced a new trajectory much more precisely.

During experiments some technological features of the RiS's representation on the screen of the display were examined also. One of them concerns the choice of the necessary image's scale.

Obviously, that it's necessary to have a field of sight providing a capture of nearby RiS's sections. The required field of sight angle grows with increase of trajectory's curvature and RiS's cross section dimension. The limiting value of this angle can be approximately determined as $\varphi_{min} = 2ar\cos\left(1-0.5LR^{-1}\right)$, where L (B or H) is RiS's dimensions, and R is its curvature radius (Fig.11). For example, at the RiS's dimensions equal to 100*m* and the curvature radius of 300*m* the limiting value of this angle makes about 70°.

However, at small sizes of the monitor's screen it can be observed only the narrow sector of environmental space, approximately ±10° (see Fig.12a). Therefore, already at small angular mismatches the RiS vanishes from a field of sight. Besides, even at exact movement along the RT with such narrow sector of view the RiS's cross section at significant RT's curvature is not seen either (pilot watches through the external wall without capture of internal one). It makes impossible further tracking of RT. Therefore to ensure the possibility of

comfortable tracking the sector of view was chosen from the condition of close-laying RiS's cross sections capture. For vigorous trajectories it made at least ±45° (see Fig.12b).

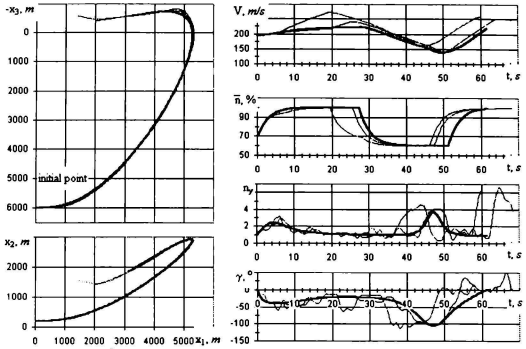


Figure 10 - Stationary point-target attack approach

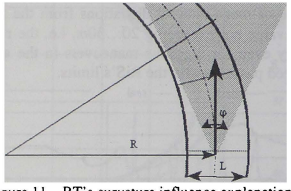
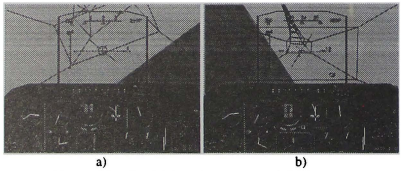


Figure 11 – RT's curvature influence explanation



a) b)

Figure 12 – Real (a) and compressed (b) images

During experiments the various images of the RiS (tunnel and gutter) were tested also. The majority of the pilots preferred the gutter, explaining it by the negative psychological reaction on "upcoming" of the top rod of RiS's skeleton in case of tunnel. Its absence in the gutter case did not worsen RiS's informativeness and did not influence significantly on the RT tracking accuracy.

III.2. Tracking of NO spatial trajectories during the landing approach

Fig.13 demonstrates the example of spatial "straight off" landing approach trajectory (without glissade gliding) and results of its realizations by the pilots during corresponding experiment. The recommended trajectory provided time-minimum exit of the a/c from the arbitrary initial conditions into the point of prelanding alignment beginning (altitude of about $10m$ and glide angle of 3°). The trajectory assumed the release of the wing mechanization and landing gear at achievement of airspeed of $400km/h$. A relay throttle switching from the maximal to the minimal thrust in some optimized instant of time was also expected. It's characteristic, that the most part of time the engines worked on the "Small gas" mode for ensuring of decrease of the large initial a/c's speed ($720km/h$).

RiS's cross section dimensions for this type of trajectories smoothly changed from 100x50m at the initial moment up to 20x2m in the terminal point.

The averaged for all received realizations results of RT tracking accuracy occurred to be as follows:

$$\begin{aligned} &I_D = 11.4 \pm 1.1m, && I_{|\Delta n_x|} = 0.093 \pm 0.009, \\ &I_{\Delta V} = -20.2 \pm 9.7m/s, && I_{\Delta\gamma} = 2.5 \pm 1.0^\circ, \\ &I_{\Delta n_y} = 0.0 \pm 0.002, && I_{|\Delta\gamma|} = 4.0 \pm 0.3^\circ, \\ &\Delta T = 20.3 \pm 12.4s. \end{aligned} \qquad (5)$$

The final errors of an exit on the runway made the following values

$$\begin{aligned} &\Delta V_f = 5.5 \pm 8.4m/s, \\ &\Delta x_{2f} = 1.8 \pm 3.0m, && \Delta x_{3f} = 4.1 \pm 6.5m, \\ &\Delta\Theta_f = 1.7 \pm 0.5^\circ, && \Delta\varphi_f = -0.5 \pm 0.5^\circ \end{aligned} \qquad (6)$$

(Θ here is the flight path angle, whereas φ is the path angle). Of course they did not affect on the subsequent landing Moreover, it is possible to ascertain with confidence that the exit in required final conditions was provided with exclusive accuracy!

The feature of RT tracking in this case was that the pilots purposely released the chassis and wing mechanization earlier than it was necessary (ordered by PSS). The release was carried out on the airspeed about

450*km/h* instead of recommended value of 400*km/h* (it's seen on the speed vs. time history at characteristic break in the speed derivative). Naturally this caused the deviation of the current speed from its prescribed value. So pilots, seeing the fall of the speed, independently interfered into the thrust control, maintaining its value close to habitual for him for this regime of flight (pilots did not have any direct information about required speed in this case). Consequently the deviation of speed, which mean integral value has made -20.2*m/s*, resulted in increase of maneuver time in the average for 20.3 seconds.

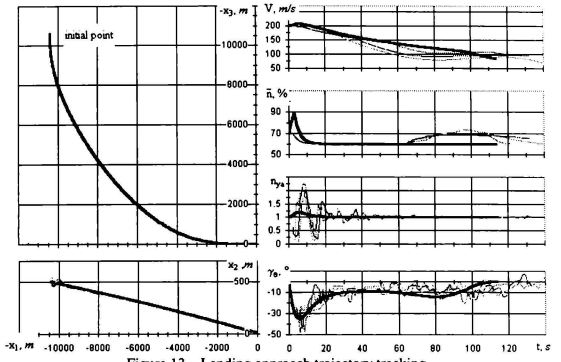


Figure 13 – Landing approach trajectory tracking

It should be noticed also, that for particular case shown on Fig.13 the fluctuations in the vertical plane at the beginning of the maneuver were caused by the intentionally entered disturbance – a/c's longitudinal balancing absence. After elimination of these fluctuations the pilot removed efforts from the stick by the trim effect mechanism and then easily traced the RT within the frame of RiS limits.

Fig.14 presents the RiS's borders and the deviations from the RT in the longitudinal and lateral direction, as well as the longitudinal and lateral stick's movements. From this data analysis it's seen that the control actions have a clear expressed adjusting character either. As earlier it could be confirmed by corresponding correlation coefficients, which are the same as in (1).

Besides a natural gradual deviation of the stick on itself in process of speed decrease is clearly seen.

The absence of additional psycho-physiological load permits the pilot to carry out other tasks, for example the maintenance of the device airspeed constant.

The results of the spectral analysis of longitudinal and lateral stick deviations are also given on Fig.14. They show that the basic frequencies of control actions are concentrated, as well as in the case of attack approach maneuver, below 0.7*Hz*.

From Fig.14 it's also seen, that NO RT in this case provides enough large $\Delta\tau_c$ - not less than 5-10 seconds. This time reserve due to the small height of the narrowed corridor strongly decreases at final part of RT. The last of course requires more pilots' patience.

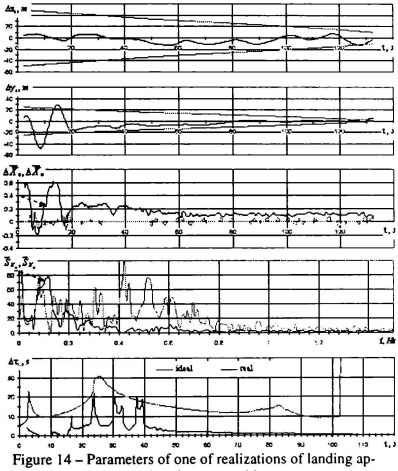


Figure 14 – Parameters of one of realizations of landing approach trajectory tracking

On the whole, according to results of objective data and pilots interviewing, the presence of RiS assistance considerably facilitates the performing of prelanding maneuver and provides more accurate exit to the landing point. But for sure pilots need extra information on

current required speed. (A.Titov suggested to use and pretested the kinestatic throttle for this purpose.)

Especially it's worth to note that the majority of the pilots has expressed a readiness in case of necessity at presence of PSS (certainly, under condition of its reliability) to leave the standard scheme of landing approach ("on the box") or even to transfer a control functions to automatics!

At real application of PSS onboard an a/c the altitude of terminal point of RT should be limited from a condition of flight safety due to the accuracy of a/c's position measurement. Therefore basically the RT should provide an exit of a/c not to the runway edge point but somewhere on glissade (on necessary range to the runway), providing the further gliding along glissade. During glissade tracking the RiS's cross section dimensions are well known (Table 2). The glissade has 3° glide angle and passes above runway edge at altitude of $15m$.

Table 2.

Altitude, *m*	RiS's width, *m*	RiS's high, *m*
400	190	82
100	112	30
60	74	24
30	38	10

In this experiments the pilots also have given the high positive ratings to the new director indication at the flight along glissade. The cognitive representation of glissade as the RiS's corridor gives the pilots the complete information on a/c respective position. According to them the tracking of submitted RiS is easier, than the constant observation of the course-glissade lines at usual (existing) directors.

The application of additional marker indicating the flight direction allowed increasing the accuracy of the RiS tracking at rectilinear trajectory. The pointing of this marker on the following RiS's cross section provided the movement along glissade without the exits for RiS's borders. The necessity of strict tracking of altitude as recommended by the RT requires the tunnel image of RiS. The straightforwardness of the trajectory allows using here the real scale of the image even at the limited sizes of the display's screen. Thus because of narrow sector of view, only remote sections of RiS's corridor, without the nearest sections is seen. Being guided by them pilot flies through the corridor on prospect that raises the accuracy of RT tracking. However in case of essential deviations it is possible to lose the RiS from the field of vision. Therefore the increase of vision-sector in this case also can be justified.

Fig.15 shows the example of realization of landing approach on RNP tunnel in conditions of a lateral wind of $10m/s$. At the moment of wind impulse from the left side occurrence (on the 22nd second) the a/c reacts to arisen sliding on the left console by the roll to the right side. Respectively pilot tries to remove it by the correspondent deflection of the stick. In the further a/c eliminates the sliding, getting a bearing angle of about 5°. The pilot pointing the flight direction marker along the RiS provides the movement inside the corridor by easy adjusting stick movements. Operating by throttle he provides the required decrease of the speed. The bearing angle elimination is being made at the earth approach already after passing the last RiS section.

The landing with shift wind and strong atmospheric turbulence of the near-earth layer was tested in the similar way.

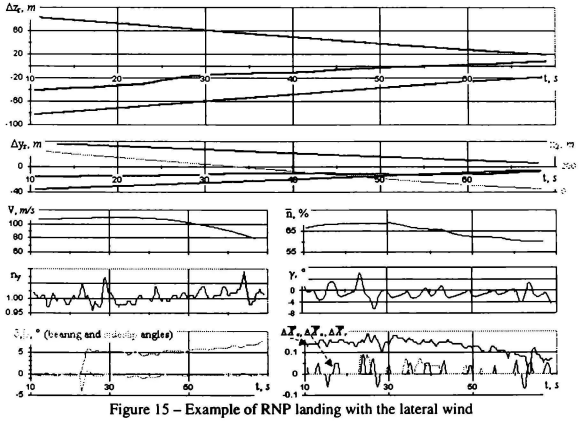


Figure 15 – Example of RNP landing with the lateral wind

III.3. The modeling of PSS application on the stages of air combat, climbing and flight on a route

To research the application of PSS's algorithms during air combat the pilots were offered for tracking the trajectories of guidance on the air target, which were updated every 5*s* due to the maneuvering of the target. The minimum-time trajectories provided the exit of the a/c into the region of effective use of appropriate armaments (gun or guided missiles).

Averaged for all received trajectories results on tracking accuracy occur to have the following values:

$$I_D = 35 \pm 4m, \qquad I_{|\Delta n_y|} = 0.66 \pm 0.44,$$
$$I_{\Delta V} = -29.3 \pm 8.6 m/s, \quad I_{\Delta\gamma} = -11.0 \pm 2.8°, \qquad (7)$$
$$I_{\Delta n_y} = -0.16 \pm 0.05, \quad I_{|\Delta\gamma|} = 16.9 \pm 3.9°.$$

So as earlier they show the affinity of received trajectories to the recommended ones.

As it's seen these meanings are close to the values obtained at the tracking of vigorous trajectories of surface-based target attack approach; and confirm the high accuracy of these RT observation. Small (about 2*s*) increase of the maneuver time in comparison with its reference value in this case did not complicate the attack of the moveable air target, because on the final part of the RT a target was already in a zone of direct visibility and the pilot carried out a targeting with the help of usual onboard means.

On a whole, all pilots marked a high efficiency and necessity of RiS prompt up to the moment of armament application with the appropriate indication of its type. Especially in conditions, when the visual contact with the target is lost (the loss of the target in roundabout of close maneuverable air combat).

The results of experiments on observation of not vigorous trajectories, such as climbing and flight on a route show that the pilots carry out the flight without any additional efforts. The accuracy of RT tracking in this case is the highest. Moreover, at presence of the certain skills the pilots without any difficulties can purposely go out from the RiS and then to come back on it, though basically in accordance to PSS concept in case of large progressive errors the new RT should be calculated[5].

However, in view of the large duration of a routing flight, the pilots would like to use the automatic tracking of RT with RiS image, and to have only supervising functions. RiS image in this case is very useful because it shows both the current relative position of a/c with respect to the RT and prognosis position. So that is a good instrument to check the automatics' serviceability.

III.4. RiS's cross-section dimensions influence

As it was already mentioned above the RiS's cross-section dimensions influent essentially on the RT tracking quality.

So on Fig.16 the parameters of the test trajectory and its realizations for three variants of RiS's corridor dimensions – 50x25*m* (A), 100x50*m* (B), 200x100*m* (C) are submitted.

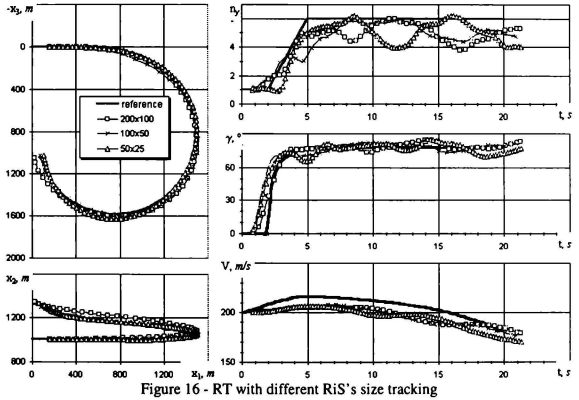


Figure 16 - RT with different RiS's size tracking

The test trajectory consisted of the following pieces: the level flight within 2 seconds, input to the turn with increase of the load factor up to 6 units within 3 seconds and further turn with the constant load factor and bank angle within 15 seconds. The magnitude of the bank angle provided the turn with a small increasing of the altitude. The maneuver assumed the maximal mode of engines without afterburning.

The quality of RT tracking for these variants of RiS's cross section sizes can be characterized by the data of

Table 3. In addition to (1) here are used the maximum absolute and relative values of an exit for the RiS's limits δ_y, δ_z, $\bar{\delta}_y$, $\bar{\delta}_z$, determined in accordance to

$$\delta_y = |\Delta y_r|_{\max} - 0.5H_r, \quad \delta_z = |\Delta z_r|_{\max} - 0.5B_r, \quad \bar{\delta}_y = 100\delta_y H_r^{-1}, \quad \bar{\delta}_z = 100\delta_z B_r^{-1}, \tag{8}$$

where A_{n_y} and A_{x_r} are the scope of fluctuations of the load factor and stick's longitudinal deviation during the turn. Some of results are also shown on Fig.17.

Table 3.

RiS's cross section size	A	B	C
l_p, m	18	29	43
$l_{\|\Delta z_r\|}$, m	10	15	35
$l_{\|\Delta y_r\|}$, m	13	21	20
$\|\Delta z_r\|_{\max}$, m	25	39	66
$\|\Delta y_r\|_{\max}$, m	41	54	77
$\delta_z = \|\Delta z_r\|_{\max} - 0.5B_r$, m	0	-11	-34
$\delta_y = \|\Delta y_r\|_{\max} - 0.5H_r$, m	28.5	29	27
$\bar{\delta}_z = 100\delta_z B_r^{-1}$, %	0	-11	-17
$\bar{\delta}_y = 100\delta_y H_r^{-1}$, %	114	58	27
$l_{\Delta n_y}$	-0.96	-0.86	-0.87
A_{n_y}	1.7	1.5	1.4
A_{x_r}	0.5	0.25	0.34

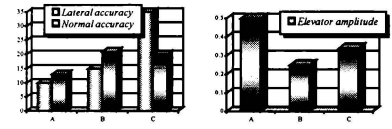


Figure 17 - Accuracy and character of pilot's control actions versus the RiS's size

The time $\Delta\tau_c$ at exact movement along the RT and fixed recommended controls during the steady turn with climbing is infinitive, as the controls are constant. But at real tracking of RT due to the presence of pilot's errors $\Delta\tau_c$ certainly is not infinitive and strongly decreases, especially at small RiS's cross section dimensions. At small dimensions, in case when the pilot goes from the road and then tries to enter it again, during the movement between RiS's walls he has not enough time to level the flight. That results in an exit for an opposite wall, i.e. in fluctuations about the RiS.

The experimental data enable to determine exactly the optimal dimensions of the RiS for each stage of flight (see also Ref.9). But at least on the basis of all experimental results obtained for such high maneuverable a/c as Su-27 "Flanker" and MiG-29 "Fulcrum" for vigorous trajectories of battle maneuvering, proceeding from the required accuracy and reliability of NO RT tracking, it's possible to limit these dimensions by the corridor of about 100m height and 100...200m width. Thus it is desirable, that the generated by PSS RT in each point provided the critical time not less than 5 seconds.

Conclusions

As results of researches show, the concept of PSS can be easily realized even onboard the modern aircraft. In all experiments the high efficiency and necessity for the pilot of such kind of support was proved. It was proved also that the RiS image carries in itself the full information, which is necessary for the a/c control. Even at small display use the application road-in-the-sky prompt gives enough good results. Obviously, that the maximal quality of the reference trajectories tracking can be achieved at visualization of the road in a real scale and wide sector of vision, for example, directly on a cockpit shied or helmet's display. Flight simulators should use for this purpose a set of displays fixed together.

References

1. Yakimenko O.A., Firsov A.V., Demidov V.V., Evfeev A.A. Mobile onboard flight simulator and pilot's preflight preparation and tactical thinking training // A collection of technical papers of AIAA Modeling & Simulation Technologies Conference, Boston, MA, 1998.
2. Firsov A.V., Yakimenko O.A. Mobile adaptive training system for pilot's on-board pre-flight preparation and tactical thinking training // Proceedings of 10th ITEC, The Hague, The Netherlands, 1999.
3. Demidov V.V., Dyshalenkov A.A., Nikitin A.P., Semenov S.S., Yakimenko O.A. High visualization capacities flight minisimulator with demonstrated prototype of operator motor action support system // A collection of technical papers of AIAA Flight Simulation Technologies Conference, San Diego, CA, 1996.
4. Yakimenko O.A. Realization of an aircraft's movement on the spatial quasi-optimal trajectories in director control mode with sight within the framework of the pilot's intellectual support system // Airspace techniques and engineering, 1998, №4.
5. Yakimenko O.A. Shortcut-time spatial trajectories onboard optimization and their cognitive head-up display visualization for pilot's control actions during maneuvering support // 17th ICIASF'97 record, Pacific Grove, CA, 1997.
6. Yakimenko O.A. Simplified modification of direct variational method of trajectory boundary optimizational tasks solution on the board of air vehicle // Russian Academy of Sciences News. Theory of Control Systems, 1998, №3.
7. Dobrokhodov V.N., Yakimenko O.A. Airplane trajectory control at the stage of rendezvous with maneuvering object algorithms synthesis // Ibidem, 1999, №2.
8. Alekhin D.V., Yakimenko O.A. The synthesis of algorithm of route flight trajectory optimization on the basis of direct method of variational calculus modification in interests of pilot intellectual support system // Ibidem.
9. Theunissen E. Integrated design of a Man-Machine Interface for 4-D Navigation. PhD Thesises. - Delft University Press, 1997.

SPATIAL FREQUENCY AND PLATFORM MOTION EFFECTS ON HELICOPTER ALTITUDE CONTROL

Jeffery A. Schroeder*
NASA Ames Research Center
Moffett Field, CA

William W. Y. Chung**
Logicon Information Systems & Services
Moffett Field, CA

Ronald A. Hess†
University of California, Davis
Davis, CA

Abstract

An experiment examined how visual scene and platform motion variations affected a pilot's ability to perform altitude changes. Pilots controlled a helicopter model in the vertical axis and moved between two points 32-ft apart in a specified time. Four factors were varied: visual scene spatial frequency, visual scene background, motion filter gain, and motion filter natural frequency. Drawing alternating black and white stripes of varying widths between the two extreme altitude points varied visual scene spatial frequency. Visual scene background varied by either drawing the stripes to fill the entire field-of-view or by placing the stripes on a narrow pole with a natural sky and ground plane behind the pole. Both the motion filter gain and natural frequency were varied in the motion platform command software. Five pilots evaluated all combinations of the visual and motion variations. The results showed that only the motion filter natural frequency and visual scene background affected pilot performance and their subjective ratings. No significant effects of spatial frequency or motion system gain were found for the values examined in this tracking task. A previous motion fidelity criterion was found to still be a reasonable predictor of motion fidelity.

Notation

g	gravitational acceleration, ft/sec^2
$h, \dot{h}, \ddot{h}$	altitude, rate, and accel., ft, ft/sec, ft/sec^2
$\ddot{h}_c$	motion platform commanded accel., ft/sec^2
h_{des}	desired altitude, ft
HQR	handling qualities rating
K	motion filter gain
n	number of data points in each mean
p	probability that effects are random
s	complex variable, rad/sec
δ_c	collective lever position, in
ω_n	motion filter natural frequency, rad/sec

Introduction

While previous efforts have suggested and examined the requirements for helicopter flight simulators, it is acknowledged that much remains unknown.[1-7] In particular, if a simulator user wants to know precisely what visual and motion cues are needed to represent an in-flight task satisfactorily, rules of thumb are available based on experience. Only sparse data are available for their support. This situation has led to continuing controversy over the role of motion platforms, g-seats, texture, field-of-view, and many other visual and motion characteristics that contribute to simulator fidelity.

A previous study addressed motion platforms by developing a fidelity criterion in the vertical axis.[8] However, systematic visual scene variations were not examined in that study. Since it is known that motion perception is affected strongly by the visual scene,[9] it is natural to believe that motion platform requirements need to be a function of the visual scene characteristics. The purpose of this study was to determine if the previous motion platform criterion depends on an easily

* Aerospace Engineer. Senior Member AIAA.
** Aerospace Engineer. Member AIAA.
† Professor. Associate Fellow AIAA.

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner.

manipulated visual characteristic: visual spatial frequency. The number of repeating patterns per degree of visual angle measures this characteristic.[10]

With a realistic helicopter model, five test pilots performed rapid 32-ft altitude ascents and descents with six motion and six visual conditions. Included in the motion conditions was one in which the motion platform moved the same amount as the visual scene. In each visual scene, constant-width horizontal black and white stripes were presented either across the entire field-of-view or on a board superimposed on a natural background. Stripe width was varied across the visual configurations. Pilots evaluated the handling qualities, motion fidelity, and the susceptibility to pilot-induced oscillations for all combinations of the visual and motion variations. Both objective and subjective data were analyzed.

Previous Relevant Research

To control altitude in a hovering helicopter, pilots likely use many feedbacks,[11] but they at least close the two outer loops shown in Fig. 1. The feedback of vertical rate-of-climb is important, as pilots are basically controlling vertical acceleration with collective position, and the feedback of only altitude to a collective command would result in a poorly damped closed-loop system.

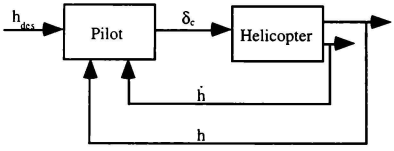


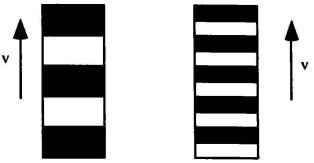
Figure 1 – Outer loops in altitude control

Visual research. Pilots must determine rate-of-climb from the available simulator cues. Although previous research has shown that estimates of rate-of-climb improve with addition of platform motion,[12] a variety of visual cues from the simulated scene predominate in the determination of speed, or here, rate-of-climb. Two of the most studied cues are visual flow rate and visual edge rate.[13-17]

Visual flow rate is the angular rate that an object moves in the visual scene. It is proportional to speed and inversely proportional to the distance from the contour or object. The number of contours or objects that pass by in a given time measures visual edge rate. It is also proportional to speed, but it is inversely proportional to the object's size.

Studies have shown that some people inappropriately use edge rate when flow rate would be more appropriate,[13] some are more sensitive to one or the other,[14] some adapt over time,[15] and that the ability to switch between the two may not be consciously possible but may depend on unconscious perceptual skills.[16] The studies agree that both cues contribute towards speed and acceleration perception, but edge rate may have a more pronounced effect.[17]

Figure 2 shows two objects placed at the same distance in front of an observer, and each object moves upwards with a speed v. Object 2 has twice the spatial frequency of object 1, since its pattern repeats twice as often per degree subtended at the observer's eye. For the same speed, v, the edge rate provided by object 2 is twice that of object 1, but the flow rates provided by both are the same.



Object 1 Object 2

Figure 2 – Spatial frequency example

If observers gaze separately at one of the above objects, previous research indicates that the perceived velocity of object 2 will be greater than that of object 1.[18] However, the perceived increase in velocity depends on the increase in spatial frequency. That is, object 2 will not seem to move twice as fast as object 1, even though the spatial frequency has been doubled. Specifically, Ref. 18 found a 31% increase in perceived velocity when the spatial frequency was doubled from 0.016 to 0.033 cycles/deg. When the spatial frequency was quadrupled (from 0.016 to 0.066 cycles/deg), the perceived increase in velocity was 240%. An additional result is that if the two objects moved at the same speed and had the same spatial frequency, the one covering the larger field-of-view seems to move slower.[18]

The above results are for a fixed gaze. If, instead, the eyes track the moving object, the perceived speed is less than with the eyes fixed. The disparity between the eye-tracked and the eye-fixed perception decreases as the spatial frequency decreases.[19] In fact, with an object consisting of a single edge, the disparity disappears.

Platform motion research. The effects of vertical platform motion on helicopters for tracking and disturbance rejection tasks were examined by Bray.[4] He concluded that the phase fidelity of the vertical acceleration cues should be accurate down to 1.0–1.5 rad/sec. This conclusion was reached using several qualitatively different natural scenes, including tracking over a runway and behind a target aircraft.

Sinacori suggested a useful translational platform motion criterion based upon considerable experience by him and others in the simulation field.[3] That criterion compares the accelerations provided to the pilot by the motion platform against those produced by the simulation mathematical model. Taking a 1 rad/sec math model acceleration as an input and the simulator platform

acceleration at 1 rad/sec as the resulting output, the relative attenuation and phase shift between these two signals is plotted.

Figure 3 shows such a plot with the six platform motion configurations examined in this study, as later discussed. Sinacori conducted a limited study to validate his suggested fidelity boundaries.[3] Reference 7 documents a detailed validation of the criterion and suggested modifications to the fidelity boundaries, which are those of Fig. 3. However, systematic variations in the visual cues were not made during the criterion's validation.

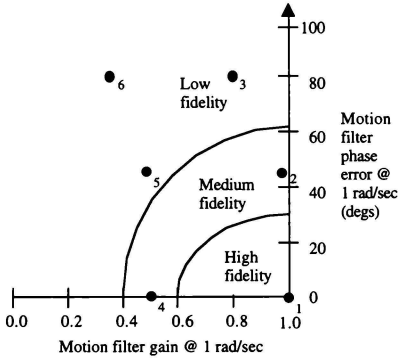


Figure 3 – Vertical motion criterion

Summarizing the above previous visual and motion efforts, pilots use rate-of-climb to control altitude. Many different visual cues contribute towards their rate-of-climb estimate. Visual edge rate, which is directly proportional to spatial frequency, is an easily manipulated visual cue and seems to provide profound effects on velocity perception. Platform motion criteria have received considerable attention, but without systematically varying the visual cues. This experiment manipulated visual spatial frequency to determine if it affects the validity of a previous vertical motion criterion.

Experiment Description

Simulator Subsystems

Vehicle Math Model. The model was intentionally simplified to have only a vertical degree-of-freedom. This allowed pilots to focus on the vertical cues. The rate-of-climb transfer function was:

$$\frac{\dot{h}}{\delta_c}(s) = \frac{14.6(s+4.82)}{(s+0.122)(s+12.9)} \qquad (1)$$

This transfer function, with an additional 60 msec of equivalent time delay as discussed later, represents a credible model of the AH-64 Apache in hover.[20] The heave damping root at –0.122 is augmented by a lead-lag that approximates the effect of dynamic inflow. All states in the other vehicle degrees-of-freedom remained zero.

Motion System. The vertical axis of the NASA Ames Vertical Motion Simulator (VMS), shown in Figure 4, was used. Reference 21 gives a detailed description of the simulator and facility. The large vertical travel of the VMS allows pilots to fly reasonable altitude-repositioning tasks without any motion cue attenuation.

The phase response of the vertical-axis servo dynamics matches an equivalent time delay of 120 msec. As discussed above, eqn. 1 needs an additional 60 msec of time delay in order to be representative of the AH-64. Rather than insert this additional 60 msec in the math model, it was more than subsumed by the 120 msec of motion system delay. Thus, the visual and motion cues in the vertical axis of this AH-64 simulation had 60 msec more delay than would be expected in the actual vehicle. This amount of additional delay is within that allowed for FAA helicopter training simulators.[1]

VMS Nominal operational motion limits

Axis	Displ	Velocity	Accel
Vertical	± 30	16	24
Lateral	± 20	8	16
Longitudinal	± 4	4	10
Roll	± 18	40	115
Pitch	± 18	40	115
Yaw	± 24	46	115

All numbers, units ft, deg, sec

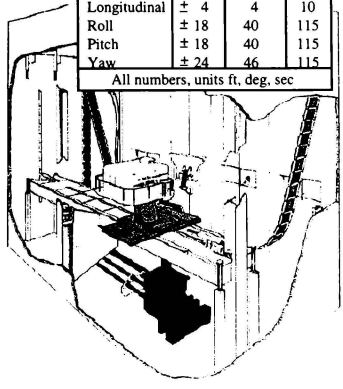


Figure 4 - Vertical Motion Simulator

Figure 5 shows the motion platform drive law. A high-pass, or washout, filter calculates the command to the motion system from the vertical acceleration of the math model. Setting the gain, K, and natural frequency, ω_n, of the filter (the filter's damping ratio is often left fixed) controls the amount of platform motion used during a simulation. Less motion is used as K decreases and ω_n increases. The predicted fidelity effect of changes in these two parameters may be found by finding the gain and

phase shift of this filter at 1 rad/sec and plotting the result on the Fig. 3 criterion.

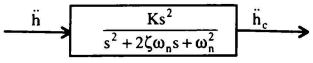


Figure 5 – Motion platform drive law

Visual System. The experiment used the Evans and Sutherland ESIG 3000 image generator. A pure time delay was added to the nominal visual system time delay so that the total visual delay from stick input to image refresh was 120 msec. Thus, the visual and motion time delays were effectively identical.

The VMS's RCAB was used, but the visual scene was presented on only the top three windows (the chin window was not used in order to prevent the ground from being viewed in the "no background" visual case discussed later). The horizontal field-of-view spans ±78 degrees, and the vertical field-of-view spans –16 to 12 degs for the upper three windows, as shown in the Fig. 6 Hammer chart. This available field-of-view is less than that in most actual helicopters.[4]

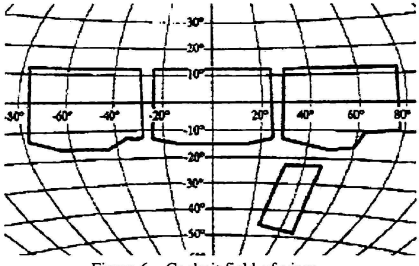


Figure 6 – Cockpit field-of-view

Cockpit. The only cockpit control was a left-handed collective from a UH-60 Black Hawk. This collective was freely moving (not spring loaded), with static friction that could be adjusted by the pilot. No instruments were present, so that the pilot had to extract all cues from the motion system, the visual system, and the inceptor characteristics.

Subjects. Five pilots participated in the study. Four of the pilots were rotorcraft test pilots with extensive experience. The fifth pilot had minimal helicopter experience, but significant fixed-wing jet transport experience. Four pilots were from NASA Ames, and one from the U.S. Army at Fort Rucker.

Task and Procedures

The task consisted of four segments, as illustrated in Fig. 7. Pilots started at a 45-ft hover 150-ft in front of a 4-ft diameter red disk. The first segment was a 32-ft ascent to the 4-ft diameter red disk at the top of the diagram. Pilots pushed a button with their right hand when initiating this first segment. After placing a set of crosshairs fixed on the canopy within the red disk, and when confident that the crosshairs could be maintained within that disk, the pilots again pushed a button to start the second task segment. In this segment the pilot had to stabilize for 10-sec, keeping the crosshairs within the red disk. The third segment was a 32-ft descent to the bottom red disk, again with button pushes beginning and terminating the reposition. The final segment was a 10-sec stabilization keeping the crosshairs within the bottom disk.

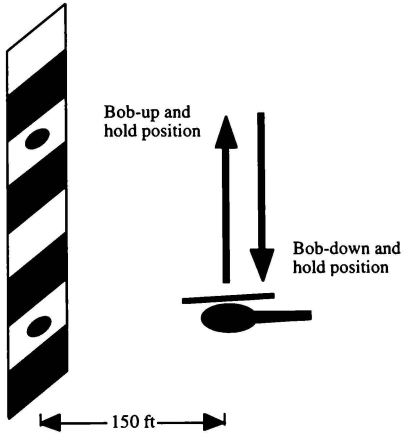


Figure 7 – Altitude repositioning task

The performance standards for each segment are given in Table 1. A set of colored lights on top of the cockpit panel indicated the performance to the pilots for each task segment. After completing the task, pilots assigned a Cooper-Harper handling qualities rating,[22] a Motion Fidelity rating (using the definitions in Table 2), and a Pilot Induced Oscillation (PIO) rating.[23]

Table 1 – Task performance standards

Segment	Desired	Adequate
Ascent	< 6 sec	< 10 sec
10-sec at top	± 2 ft	± 5 ft
Descent	< 6 sec	< 10 sec
10-sec at bottom	± 2 ft	± 5 ft

Table 2 - Motion Fidelity Scale

Fidelity Rating	Definition
High	Motion sensations like those of flight.
Medium	Motion sensations are noticeably different from flight, but not objectionable.
Low	Motion sensations are noticeably different from flight and objectionable.

Configurations

The two variables K and ω_n for the motion filter of Fig. 5 were varied as shown in Table 3. The filter's damping ratio was held constant at 0.7. These values were selected to span a range of predicted motion fidelities per the criterion in Ref. 7, and each configuration number is plotted on Fig. 3. These predicted fidelities apply to tasks that have both tracking and disturbance rejection components. Here, no disturbances were present, so the task involved only tracking. For tracking-only tasks, previous work has shown that the predicted motion fidelity is affected by motion filter natural frequency, rather than by motion filter gain.[4,7] As such, the predicted fidelity for the configurations with K=0.5 would be expected to be that for K=1.0 at each corresponding natural frequency. This is reflected in the last column of Table 3.

Table 3 – Motion filter configurations

Config	K	ω_n	Predicted fidelity from Ref. 7	Predicted fidelity for tracking only
1	1.0	0.00	High	High
2	1.0	0.52	Medium	Medium
3	1.0	0.89	Low	Low
4	0.5	0.00	Medium	High
5	0.5	0.52	Low	Medium
6	0.5	0.89	Low	Low

The amount of platform travel required for particular filter depends upon the frequency content of the task. For the full motion configuration (K=1.0, ω_n=0.00), if a pilot stays within the desired performance boundaries for the altitude reposition, the required platform travel would be 36 ft (32 ft between the two points plus 2 ft of allowed error at both ends). The amount of platform motion required for the other two configurations for which K=1 was determined by randomly sampling 10 runs for each motion configuration and finding the range of platform motion used in each during the task. The mean and standard error of the travel required for the K=1, ω_n=0.52 configuration was 19.0±0.9 ft and for the K=1, ω_n=0.89 configuration was 12.4±0.8 ft. When the gain, K, drops to 0.5, one-half of the above travels would be expected in each case.

Three stripe widths were tested: 2 ft, 8 ft, and 32 ft. The vertical plane containing the stripes was 150 feet in front of the pilot's eyes. These widths correspond to spatial frequencies of 0.65, 0.16, and 0.041 cycles/deg, respectively, as measured directly in front of the pilot's eyes.

The stripes were presented without and with a background, as shown in Figs. 8 and 9. The red disks, representing the ascent and descent end points, and the crosshairs are also shown in both figures. The position of the crosshairs indicates that the aircraft is centered between the two red disks. The numerals on Figs. 8 and 9 were not shown to the pilot.

Without the background, the stripes covered the entire field-of-view across all three windows. With a background, the stripes were on a vertical board 4 feet wide. A sky and flat textured ground plane, including a runway (giving some familiar size cues), was behind this vertical board. The sky and ground plane projected into both side windows. Since the horizon is always level with the pilot's eyes, it cuts through the vertical board at that particular eye height. This provides a very compelling height cue. Also, with the background, the visual scene was more natural appearing, and it was chosen as a configuration to break the monotony of flying against the laboratory-like scene without the background.

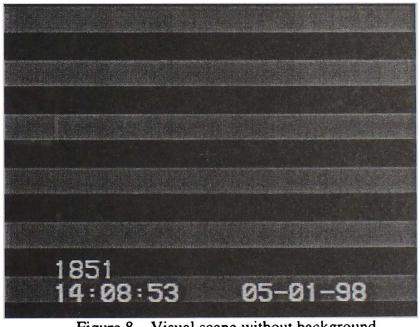


Figure 8 – Visual scene without background

Even with this simple background scene, many additional visual cues for altitude and altitude rate become available. These cues include the additional visual flow rate from the contours on the ground, and angular size changes of the ground polygons, and the interposition cues arising from areas behind the vertical board appearing and disappearing during altitude changes. Reference 24 reported that pilots make effective use of the depression angles to lines that are orthogonal to the forward gaze in order to control altitude.

Specifying these cues quantitatively is more difficult than in the laboratory-like scene without a background, and no attempt was made to do so. Here we can only determine

whether or not these additional cues had an effect on pilot performance or opinion.

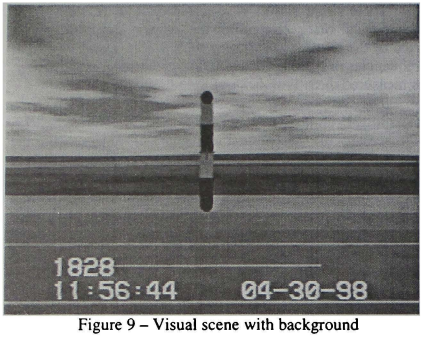


Figure 9 – Visual scene with background

Each of the five pilots evaluated the above four factors, which combine to give 36 configurations (2 motion filter gains x 3 motion filter natural frequencies x 3 spatial frequencies x 2 backgrounds). The configurations were randomized and unknown to the pilots.

Results

Both objective and subjective data were analyzed. The objective data consisted of the time to ascend and descend between the targets and the maximum error at the top and bottom targets. These four measures were examined, since the pilot based the performance part of his subjective handling qualities rating on these four measures (Table 1). The subjective data analyzed consisted of the handling qualities ratings, motion fidelity ratings, and pilot induced oscillation ratings.

To determine how both the objective and subjective measures depended upon the experimental manipulations, a repeated measures analysis of variance was performed.[25] Only the statistically significant results ($p<0.05$) are reported. This means that the odds of being incorrect in saying a particular measure was affected by a particular manipulation (rather than the difference being due to chance) is less than one in twenty.

All of the subsequent plots are in a consistent format in that means and error bars indicating the standard error of the mean are plotted. The standard error of the mean provides a confidence band around the experimentally determined mean. The probability that the true mean of the entire pilot population lies outside twice the error bars is less than one in twenty.

Objective Performance Data

Tracking error at upper disk. Motion filter natural frequency and scene background each affected tracking error at the upper disk. No interactions among the manipulations occurred.

Figure 10 shows the means bounded by the standard errors of the mean for the maximum tracking error at the upper disk as the motion filter natural frequency varied ($p=0.003$). Here each mean is determined from the runs with motion filter natural frequencies of 0.00, 0.52, and 0.89 rad/sec, respectively, regardless of the other manipulations, since no interactions were found in the statistical analysis. So, it can be said that the motion filter natural frequency affected tracking error independent of the other variables in the experiment. As indicated, there are 60 points used for each mean (5 pilots x 2 motion filter gains x 3 spatial frequencies x 2 scene backgrounds).

All means were in the ±2 ft desired performance range about the center of the disk (Table 1). Tracking error degraded as the natural frequency of the filter increased. This is consistent with previous results, which have shown that the distortion introduced by the motion filter natural frequency reduces the phase margin of the tracking loop, which in turn negatively impacts the closed-loop damping ratio of the pilot-vehicle system.[4,7] More overshoots occur, and larger maximum tracking errors then follow.

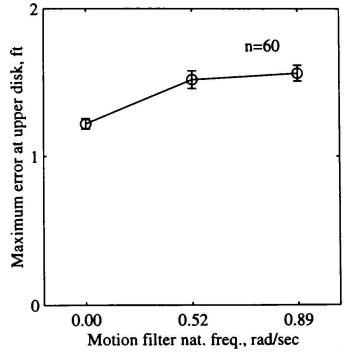


Figure 10 – Effect of motion filter ω_n on upper tracking error

Figure 11 shows that the presence of a visual background also influenced tracking error ($p=0.017$). Although having the stripes span the entire field-of-view provided extremely precise altitude and altitude rate information, pilots performed slightly worse overall without the background versus with it. Although Ref. 13 showed that perceived speed becomes slower as the stimulus field becomes larger, more than just stimulus field size varied here. As discussed earlier, pilots also received additional altitude and speed cues from having the natural background. In addition, the presence of the horizon line always showing the pilot's height relative to

the vertical board was likely a compelling and precise altitude cue.

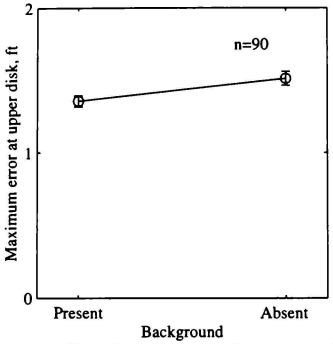


Figure 11 – Effect of scene background on upper tracking error

Tracking error at lower disk. Motion filter natural frequency was a main effect for the tracking at the lower target. However, in addition, a two-way interaction occurred among motion filter natural frequency and background presence.

Figure 12 shows the effect of motion filter natural frequency (p=0.003). Again, tracking error degraded as motion filter natural frequency increased as with the upper disk. The interaction is shown in Fig. 13 (p=0.031), but the interaction does not appear particularly strong. The error appears to perhaps be worse in going from 0.00 to 0.52 rad/sec with the background scene than without the background and vice versa when going from 0.52 to 0.89 rad/sec.

One may ask why motion filter gain did not affect the tracking error in this experiment. The fact that no gain effects were revealed is consistent with previous results in Refs 4 and 7. Since this was a tracking task in which the pilot generates all of his or her motion, the effect of motion filter gain is not an influence for vehicles with reasonable control sensitivities. It is when a disturbance rejection task is added, in which the pilot does not generate all of the motion, that motion filter gain has an effect.[4,7]

As to why spatial frequency did not have an effect on tracking error is unknown. Perhaps a large enough range in spatial frequency was not examined. Or, pilots may have been able to extract sufficient velocity cues using the angular rate of the edges as opposed to using the edge rate of the objects.

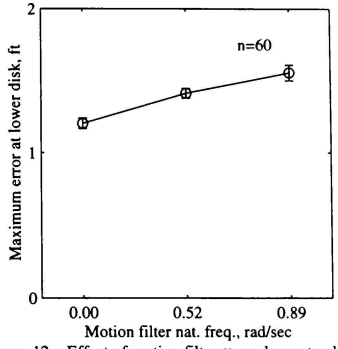


Figure 12 – Effect of motion filter ω_n on lower tracking error

The other two objective performance measures in the task were time to ascend and descend between the targets. No statistically significant effects were found in going to the upper target, but a three-way interaction among motion filter gain, motion filter natural frequency, and spatial frequency occurred when going to the lower target (p=0.005). Plots of this complex interaction did not reveal any interpretable trend.

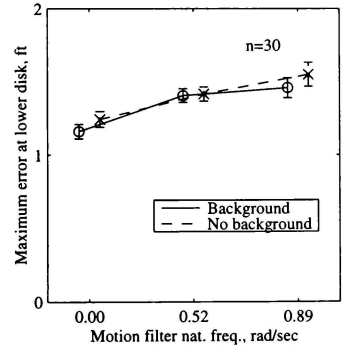


Figure 13 – Effect of motion filter ω_n and scene background on lower tracking error

Subjective Performance Data

Handling qualities ratings. Motion filter natural frequency and the visual background again were the main effects. Also, a three-way interaction among motion filter gain,

motion filter natural frequency, and visual background occurred (p=0.032).

Figure 14 shows that the motion filter natural frequency affected handling qualities ratings (p=0.001). As the natural frequency increased, the handling qualities degraded one point from just over 3 to slightly over 4. Only the natural frequency of 0.00 rad/sec received Level 1 ratings on average.

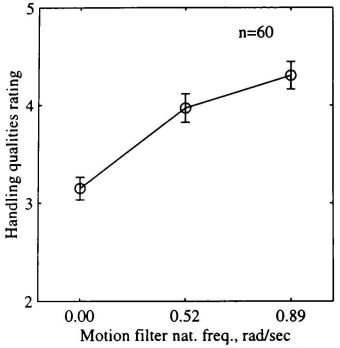


Figure 14 – Motion filter ω_n effect on HQRs

Figure 15 shows that HQRs were better with the visual background than without it (p=0.014). However, the mean differences were not as great as the variations caused by motion filter natural frequency. Since desired performance, on average, was achieved by the tested configurations, the differences in HQRs were attributed to variations in the required pilot compensation.

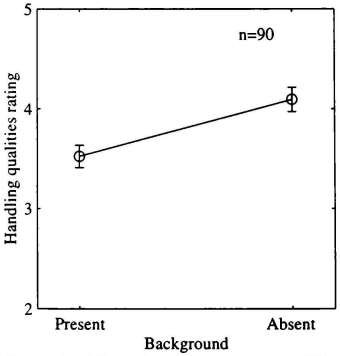


Figure 15 – Effect of scene background on HQRs

Motion fidelity ratings. As with the handling qualities ratings, motion filter natural frequency and the visual background were the main effects on motion fidelity ratings. Figure 16 shows that motion fidelity degraded as the motion filter natural frequency increased (p=0.002). A motion filter natural frequency of 0.00 resulted in a mean rating between high and medium. When increased to 0.52 rad/sec, the average rating was medium, and that rating decreased to below medium for the highest natural frequency tested. As discussed when the motion fidelities were predicted in Table 3, motion filter natural frequency is the primary determinant of motion fidelity for tracking tasks without a disturbance rejection component. For the three natural frequencies tested, one would expect the motion fidelities to be nearly high, medium, and low, for 0.00, 0.52, and 0.89 rad/sec, respectively. Here, the 0.52 rad/sec case was exactly as predicted. The 0.00 rad/sec case was on the borderline between high and medium. The 0.89 rad/sec case was worse than the 0.52 rad/sec configuration, but did not receive average ratings of low, as the criterion would predict.

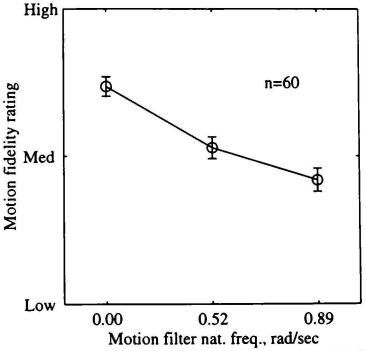


Figure 16 – Effect of motion filter ω_n on motion fidelity rating

Figure 17 repeats the criterion shown previously in Fig. 3 but with the means (and their standard error) of the data for each of the six motion configurations added. These means arise from assigning numbers to the ratings as follows: Low = 1, Medium = 2, and High = 3. So, for the averages, it is reasonable to draw dividing lines between the fidelity regions to be: Low $\leq$ 1.5, 1.5 $\leq$ Medium $\leq$ 2.5, and 2.5 $\leq$ High. Recall from the discussion of Fig. 3 that the criterion applies to tasks combining tracking and disturbance rejection. If only tracking was performed, as was the case here, then gain variations normally do not affect fidelity. Figure 17 shows this to be the case. Configurations 1, 2, 4, and 5 are in excellent agreement with the above discussion (which is effectively consistent with the discussion of Fig. 16). Data for configurations 3 and 6 did not agree

with the criterion's prediction, although configuration 6 is nearing Low fidelity. Also motion filter gain may be having a slight effect at this high level of phase distortion. So, it may be plausible to raise the Medium-to-Low fidelity border near the criterion's y-axis in light of these results.

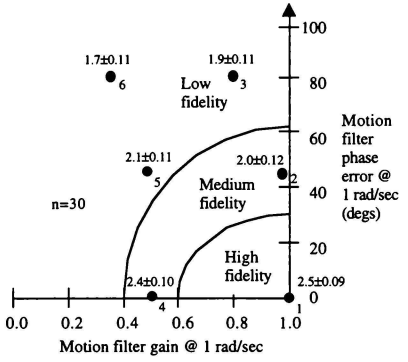


Figure 17 – Predicted versus actual motion fidelity

As shown in Fig. 18, the presence of a background improved perceived motion fidelity (p=0.046). This result is somewhat surprising. One might expect no difference between having or not having the natural visual background present, as the rating is a motion fidelity rating. Yet pilots rated the no background case as having poorer motion fidelity. Perhaps the lack of any real world objects (such as the runway) without the background made the determination of how the vehicle was actually moving more difficult. This difficulty may have affected their impression of the motion fidelity.

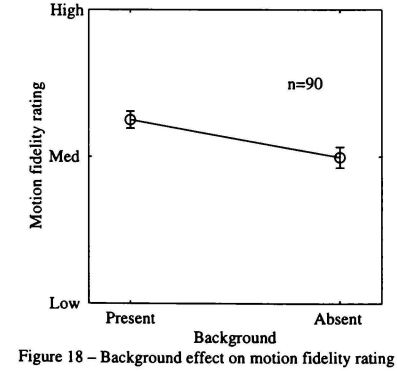


Figure 18 – Background effect on motion fidelity rating

Pilot induced oscillation ratings. Only motion filter natural frequency affected this measure, as shown in Fig. 19 (p=0.023). Less of a tendency for a pilot induced oscillation occurred for a motion filter natural frequency of 0.00 than for either 0.52 or 0.89 rad/sec. Platform motion has previously been shown to affect PIO ratings.[26] Here, poorer motion made the configurations more PIO prone.

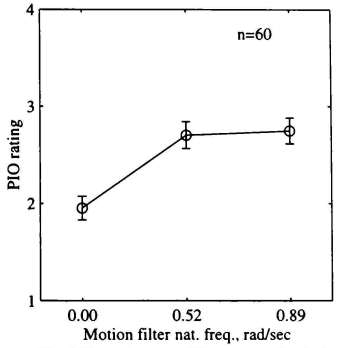


Figure 19 – Effect of motion filter ω_n on pilot induced oscillation rating

Conclusions

A piloted simulation evaluated the effects of changes in platform motion, visual scene spatial frequency, and visual scene background in an altitude control task for a helicopter. Pilots controlled only altitude and had to accurately move between two points 32 feet apart within a specified time. Six motion configurations were tested that included one configuration in which the pilot physically moved the full 32 ft required in the task. Three spatial frequencies and the presence or lack of a natural visual background were evaluated.

As motion filter natural frequency increased, tracking error, handling qualities rating, motion fidelity rating, and pilot-induced oscillation rating degraded. No statistically significant effects of motion filter gain changes were found on the above measures. These results are consistent with previous data on tracking tasks without disturbances. However, at the highest level of motion filter phase distortion tested, there was some indication the motion filter gain might be a factor.

When a natural sky-earth background was included in the visual scene, tracking error, handling qualities rating, and motion fidelity rating improved. Motion fidelity changed slightly when the background was added; however, these changes were small compared to the motion filter effects.

No effects of the spatial frequency variations were found. Since pilots were a fixed horizontal distance from the object on which the spatial frequency variations were made, perhaps they instead used visual flow rate (angular rate) of the elements in the visual scene. For a particular vehicle velocity, this flow rate does not change when the spatial frequency changes as long as a fixed distance from the object is maintained.

Overall, the motion fidelity criterion was good predictor of fidelity including these visual scene variations. This was especially the case for platform motion filters with low-to-moderate levels of phase distortion. When the motion filter had a high gain and high phase distortion, the criterion underpredicted the fidelity, and adjustment of its boundaries may be warranted.

Recommendations

Three values of spatial frequency were examined in this experiment. Future experiments should evaluate a wider range, perhaps focusing on larger, rather than smaller spatial frequencies. The lowest value tested here was 0.041 cycles/deg, and such values would seem to be easily achievable by any modern flight simulator visual system. Thus, it would seem little value would gained by testing even lower spatial frequency values.

Also, the variation in visual scene backgrounds here was qualitative rather than quantitative. Additional systematic visual scene variations should be measured and examined to determine what influence they might have on motion fidelity criteria. These effects should be examined in disturbance rejection tasks and tracking tasks.

References

1. *Helicopter Simulator Qualification*, Advisory Circular AC 120-63, Federal Aviation Administration, U.S. Department of Transportation, Oct., 1994.

2. Heffley, R.K., Clement, W.F., Ringland, R.F., Jewell, W.F., Jex, H.R., McRuer, D.T., and Carter, V.E., "Determination of Motion and Visual System Requirements for Flight Training Simulators," Technical Report 546, U.S. Army Research Institute for the Behavioral and Social Sciences, Alexandria, VA, Aug., 1981.

3. Sinacori, J.B., "The Determination of Some Requirements for a Helicopter Flight Research Simulation Facility," NASA CR-152066, Sept., 1977.

4. Bray, R.S., "Visual and Motion Cueing in Helicopter Simulation," NASA TM 86818, Sept., 1985.

5. Cross, K.D. and Gainer, C.A., "An Enumeration of Research to Determine the Optimal Design and Use of Army Flight Training Simulators," Technical Report 763, U.S., Army Research Institute for the Behavioral and Social Sciences, Alexandria, VA, Oct., 1987.

6. Boldovici, J.A, "Simulator Motion," Technical Report 961, U.S. Army Research Institute for the Behavioral and Social Sciences, Alexandria, VA, Sept. 1992.

7. Schroeder, J.A., "Helicopter Flight Simulator Motion Platform Requirements", NASA TP-208766, 1999, in press.

8. Schroeder, J.A., "Evaluation of Simulation Motion Fidelity Criteria in the Vertical and Directional Axes," *Journal of the American Helicopter Society*, Vol. 41, No. 2, Apr. 1996, pp. 44-57.

9. Young, L.R., "Visually Induced Motion in Flight Simulation," Paper 16, *Piloted Aircraft Environment Simulation Techniques*, AGARD-CP-249, Brussels, Belgium, 1978.

10. Goldstein, E.B., *Sensation and Perception*, Brooks/Cole Publishing Company, Pacific Grove, 1989, pp. 68-69.

11. Hess, R.A. and Malsbury, T., "Closed-Loop Assessment of Flight Simulator Fidelity," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 1, 1991, pp. 191-197.

12. Johnson, W.W. and Schroeder, J.A., "Visual-Motion Cueing in the Control of Altitude," IEEE Conference on Systems, Man, and Cybernetics, Vancouver, British Columbia, Oct., 1995.

13. Denton, G.G., "The Influence of Visual Pattern on Perceived Speed," *Perception*, Vol. 9, 1980, pp. 393-402.

14. Owen, D.H., Wolpert, L, and Warren, R., "Effects of Optical Flow Acceleration, Edge Acceleration, and Viewing Time on the Perception of Egospeed Acceleration," AFHRL TP-84-4, Air Force Human Resources Laboratory, Williams Air Force Base, AZ, 1984.

15. Shinar, D., Rockwell, T.H., and Malecki, J.A., "The Effects of Changes in Driver Perception on Road Curve Negotiation," *Ergonomics*, Vol. 23, 1980, pp. 263-275.

16. Johnson, W.W. and Awe, C.A., "The Selective Use of Functional Optical Variables in the Control of Forward Speed," NASA TM 108849, Sept., 1994.

17. Larish, J.F. and Flach, J.M., "Sources of Optical Information Useful for the Perception of Velocity of Rectilinear Self-Motion," *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 16, 1990, pp. 295-302.

18. Diener, H.C., Wist, E.R., Dichgans, J., and Brandt, T., "The Spatial Frequency Effect on Perceived Velocity," *Vision Research*, Vol. 16, No. 2, 1976, pp. 169-176.

19. Dichgans, J., Wist, E., Diener, H.C., and Brandt, T., "The Aubert-Fleischl Phenomenon: A Temporal Frequency Effect on Perceived Velocity in Afferent Motion Perception," *Experimental Brain Research*, Vol. 23, No. 5, 1975, pp. 529-533.

20. Schroeder, J.A., Tischler, M.B., Watson, D.C., and Eshow, M.M., "Identification and Simulation Evaluation of a Combat Helicopter in Hover," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No.1, Jan.-Feb. 1995, pp. 31-38.

21. Danek, G.L., "Vertical Motion Simulator Familiarization Guide," NASA TM 103923, May 1993.

22. Cooper, G.E. and Harper, Jr., R.P., "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities," NASA TN D-5153, Apr., 1969.

23. Smith, R.E., "Effects of Control System Dynamics on Fighter Approach and Landing Longitudinal Flying Qualities – Volume I," AFFDL-TR-78-122, Wright-Patterson Air Force Base, OH, Mar., 1978.

24. Johnson, W.W. and Phatak, A.V., "Optical Variables and Control Strategy in a Visual Hover Task," *Proceedings of the IEEE Conference on Systems, Man, & Cybernetics*, Cambridge, MA, Nov., 1989.

25. Myers, J.L., *Fundamentals of Experimental Design*, Allyn and Bacon, Inc., Boston, 1972, pp. 167–190.

26. Schroeder, J.A., Chung, W.W.Y., Tran, D.T., Laforce, S., and Bengford, N.J., "Pilot-Induced Oscillation Prediction with Three Levels of Simulation Motion Displacement," AIAA Paper 98-4333, *Proceedings of the AIAA Atmospheric Flight Mechanics Conference*, Boston, MA, Aug. 1998.

AIAA-99-4114

DYNAMIC INFLOW MODELING FOR SIMULATION OF A HELICOPTER OPERATING IN GROUND EFFECT

Hong Xin* J.V.R. Prasad†
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0150

David A. Peters‡
School of Mechanical Engineering
Washington University
St. Louis, MO 63130-4899

ABSTRACT

Adequate modeling of the ground effect phenomenon is important for rotorcraft flight dynamics analysis and flight simulation. A new approach has been developed for ground effect analysis based on the Peters-He finite-state dynamic inflow theory. The influence of the ground plane is represented as a source-like pressure perturbation in the flow field. The total pressure perturbation, which determines the induced inflow at rotor disk, is obtained as the superposition of contributions due to the rotor and the ground. Following this approach, a ground influence coefficients matrix $[G]$ is obtained relating the ground interference velocity with the rotor pressure distribution. The method is applied to both hover and forward flight conditions. The elements of $[G]$ matrix are obtained as functions of the normalized height, the tip-path-plane angle-of-attack, and the rotor wake skew angle. The structure of the $[G]$ matrix indicates coupling among the radial shape modes and azimuthal harmonics of the ground interference velocity, implying that the ground plane has an effect on the inflow distribution at rotor disk. Correlation of predicted results with experimental data indicate good over all agreement. The new model performs as well as, or better than classical image methods. However, unlike an image rotor model, the present finite-state model can easily be implemented into a flight simulation program. Results indicate that the induced power of a rotor in ground effect increases, rather than decreases, as the forward speed is increased initially from hover. Results also indicate that the fore-to-aft inflow gradient is reduced by operation close to the ground.

Nomenclature

$[A]$	Matrix defined as $\{\beta\}=[A]\{\sigma\}$
$[B]$	Matrix defined as $\{\sigma\}=[B]\{\tau\}$
C_T	Rotor thrust coefficient
C_Q	Rotor induced torque coefficient
F	Transformation function from rotor to ground ellipsoidal coordinate system
$[G]$	Ground influence coefficients matrix
h	Height of the rotor above the ground normalized with rotor radius
$\bar{h}$	Height of the rotor above the ground normalized with rotor diameter
$[I]$	Identity matrix
$[L]$	Induced inflow influence coefficient matrix
$[M]$	Apparent mass matrix
$\bar{P}_n^m$	Normalized associated Legendre function of first kind
$\bar{Q}_n^m$	Normalized associated Legendre function of second kind
q_i	i^{th} component of induced velocity
R	Rotor radius
V_m	Mass flow parameter
w	Normal component of non-dimensional induced velocity, positive downward
(x, y, z)	Cartesian coordinates at rotor disk
$(\hat{x}, \hat{y}, \hat{z})$	Cartesian coordinates at ground surface
α_{TPP}	Angle-of-attack of the rotor tip-path-plane
$\{\alpha\}$	Rotor induced inflow coefficients
$\{\beta\}$	Ground interference velocity coefficients
δ_{ij}	Kronecker delta
Φ	Non-dimensional pressure perturbation
$\Delta\Phi$	Pressure discontinuity
$\lambda, \lambda_c, \lambda_s$	Empirical data of fundamental and first harmonics of induced inflow
μ^*	Advance ratio normalized with $\sqrt{C_T/2}$
(ν, η, ψ)	Ellipsoidal coordinates at rotor disk
$(\hat{\nu}, \hat{\eta}, \hat{\psi})$	Ellipsoidal coordinates at ground surface
$\{\sigma\}$	Ground pressure coefficient
$\{\tau\}$	Rotor pressure coefficient
ξ	Coordinate along free-stream line

Superscripts

$(\,)^A, (\,)^V$	indication of unsteady, convection parts
$(\,)^c, (\,)^s$	indication of cosine, sine parts
$(\,)^m, (\,)^i$	associated with m^{th}, i^{th} harmonics
$(\,)^*$	normalized with $\sqrt{C_T/2}$

* Graduate Research Assistant, Student Member AIAA
† Professor, Senior Member AIAA
‡ Professor, Fellow AIAA

Copyright © 1999 The American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

Subscripts

$(\ )_G, (\ )_R$ associated with ground, rotor

$(\ )_n, (\ )_k, (\ )_j$ associated with n^{th}, k^{th}, j^{th} polynomial

Introduction

Modern day helicopters are frequently required to operate close to a ground plane, a building top, or a ship deck at sea. Operation of a helicopter close to the ground introduces a range of special characteristics in the flight dynamics behavior since the aerodynamic behavior of the lifting rotor in the presence of a ground plane is significantly altered. Most significant is the effect on the magnitude and distribution of the induced velocity at the rotor and hence, on the rotor thrust, hub moments and power required. Therefore, adequate modeling of the ground effect phenomenon is important for rotorcraft flight dynamics analysis and flight simulation.

In hover, the influence of ground proximity can be viewed as a reduction of the average induced velocity at the rotor disk, and thus the in-ground-effect (IGE) average inflow becomes a function of the height of the rotor above the ground. This represents a reduction in the induced power required at a constant thrust. Because of this beneficial effect, a helicopter can hover in ground effect at a higher gross weight or altitude than is possible out of ground effect. Equivalently, the ground effect increases the rotor thrust at a given power, which helps flare the helicopter when landing.

Many experimental efforts were made in wind tunnel and flight tests for understanding the basic physics of ground effect. Hayden[1] correlated flight test data to obtain an empirical formula of estimating the ground effect on the average induced velocity for a hovering helicopter. Newman[2] also suggested an empirical function from curve fitting of experimental data. These two empirical relationships are plotted in Fig. 1, where the height of the rotor above the ground is normalized with respect to rotor radius.

In forward flight, where the wake is swept behind the rotor, the effect of the ground diminishes rapidly with forward speed. Therefore, as speed is increased from hover in ground effect, power initially increases rather than decreases. Reference [3] presents wind-tunnel measurements of the power required in ground effect for a rotor $0.71R$ above the ground. By assuming that 80 percent of the hovering shaft power is induced power, the induced power vs. normalized advance ratio is plotted in Fig. 2.

Using the Princeton Dynamic Model Track, Curtiss et al[4] conducted several experiments to study the aerodynamic characteristics of a helicopter rotor operating in ground effect at low advance ratios. They also performed an empirical study to compute the inflow components using the measurements of rotor thrust and hub moments[5]. The induced velocity experienced by a rotor in ground effect was represented by constant and first harmonic terms. The influence of the ground on the longitudinal distribution of the inflow can be seen in Fig. 3, which shows the cosine (fore and aft) component of the inflow, λ_c^*, vs. advance ratio, μ^*, for varying height-to-diameter ratios, $\bar{h}$, where the superscript * represents the normalization with respect to $\sqrt{C_T/2}$.

Ground effect has been examined analytically using the method of images[6][7], where a mirror-image rotor is placed below the ground plane so that the boundary condition of no flow through the ground is automatically satisfied. In 1955, Cheeseman and Benett[6] developed an analysis for estimating the influence of the ground on the rotor lift based on the method of images. As shown in Fig. 1, Cheeseman's approach gives good agreement with empirical relations except at very low hovering height ($h < 0.4$).

Heyson[7] analyzed helicopter ground effect by representing the rotor wake using an inclined string of doublets with an image system below the ground plane. His study showed that the ground-induced interference is an upwash at rotor disk. His theoretical results are compared to the measurements of Reference [3] for uniform and triangular load distributions in Fig. 2.

Some free wake models have been incorporated to determine the realistic shape of the rotor wake in ground effect. Recent studies[8] used a numerical approach based on a combination of a free vortex wake model for the rotor and a panel method for the ground plane. The predicted radial distributions of the in-ground-effect inflow at rotor disk taken from Ref. [8] are shown in Fig. 4. However, the iterative scheme needed in the determination of the rotor wake geometry in ground effect requires too much computational effort for its implementation into a real time flight simulation program.

The 'Peters-He' finite-state dynamic inflow theory[9], also called generalized dynamic wake theory, is characterized as representation of induced inflow at rotor as dynamic degrees of freedom in a system of first order differential equations in the time domain. Due to its dynamic nature and computational efficiency, the generalized dynamic wake theory is finding a wide application in flight dynamics and aeroelasticity analyses of rotorcraft. Especially, it has been implemented in major flight simulation programs currently used in helicopter industry. However, those models use simple empirical factors to account for ground effect, which cannot accurately predict radial and azimuthal distributions of in-ground-effect induced

inflow. Thus, there is a definite need for extending the applicability of current dynamic inflow models to in-ground-effect flight conditions.

In this paper, a new approach is developed for ground effect analysis based on the generalized dynamic wake theory. The influence of the ground plane is represented as a second spatially distributed pressure perturbation in the flow field. The total pressure perturbation, which determines the induced inflow at rotor disk, is obtained as the superposition of both the rotor and the ground contributions. Following this approach, the ground effect is determined as an interference velocity distribution at the rotor disk, which is related to the rotor pressure distribution in a finite-state formulation. In-ground-effect inflow at the rotor disk is predicted and analyzed for both hover and forward flight. Comparisons are made between theoretical and experimental results.

Background and Basic Equations

The Peters-He generalized dynamic wake theory[9] was developed for an incompressible potential flow with small perturbations relative to the stream. In such a case, the continuity and momentum equations can be written as:

$$q_{i,i} = 0 \tag{1}$$

$$\dot{q}_i - V_m q_{i,\xi} = -\Phi_{,i} \tag{2}$$

where, the q_i's are the perturbation velocity components, Φ is the pressure perturbation, ( ) is a non-dimensional time derivative, and ( ),$_\xi$ is the derivative along the free-stream line. Equation (2) suggests that the pressure is the superposition of two parts:

$$\Phi = \Phi^V + \Phi^A \tag{3}$$

where, Φ^V denotes the convection part due to momentum flux, and Φ^A denotes the acceleration part due to unsteadiness. Thus,

$$V_m q_{i,\xi} = \Phi^V_{,i} \tag{4}$$

$$-\dot{q}_i = \Phi^A_{,i} \tag{5}$$

It can be shown that the total pressure as well as the individual parts satisfy the Laplace equation. When written in ellipsoidal coordinate system with origin at the rotor hub center, a suitable general solution can be obtained using the method of separation of variables. The total pressure perturbation can be written as

$$\Phi = -\frac{1}{2}\sum_{m=0}^{\infty}\sum_{n=m+1,m+3,\cdots}^{\infty}\bar{P}_n^m(\nu)\bar{Q}_n^m(i\eta)[\tau_n^{mc}\cos(m\psi)+\tau_n^{ms}\sin(m\psi)] \tag{6}$$

where $\bar{P}_n^m(i\eta)$ and $\bar{Q}_n^m(i\eta)$ are normalized associated Legendre functions of the first and second kind, respectively. Since $\bar{Q}_n^m(i0)=1$ at the rotor disk, the above expression has the following discontinuity across the rotor disk,

$$\Delta\Phi(\eta=0) = \sum_{m=0}^{\infty}\sum_{n=m+1,m+3,\cdots}^{\infty}\bar{P}_n^m(|\nu|)[(\tau_n^{mc})\cos(m\psi)+(\tau_n^{ms})\sin(m\psi)] \tag{7}$$

The normalized pressure coefficients τ_n^{mc} and τ_n^{ms} are determined by equating this pressure discontinuity to the rotor aerodynamic loads. Similarly, we have

$$\Phi^V = -\frac{1}{2}\sum_{m=0}^{\infty}\sum_{n=m+1,m+3,\cdots}^{\infty}\bar{P}_n^m(\nu)\bar{Q}_n^m(i\eta)[(\tau_n^{mc})^V\cos(m\psi)+(\tau_n^{ms})^V\sin(m\psi)] \tag{6a}$$

$$\Phi^A = -\frac{1}{2}\sum_{m=0}^{\infty}\sum_{n=m+1,m+3,\cdots}^{\infty}\bar{P}_n^m(\nu)\bar{Q}_n^m(i\eta)[(\tau_n^{mc})^A\cos(m\psi)+(\tau_n^{ms})^A\sin(m\psi)] \tag{6b}$$

and the two parts of the pressure discontinuity across the rotor disk can be written as:

$$\Delta\Phi^V(\eta=0) = \sum_{m=0}^{\infty}\sum_{n=m+1,m+3,\cdots}^{\infty}\bar{P}_n^m(|\nu|)[(\tau_n^{mc})^V\cos(m\psi)+(\tau_n^{ms})^V\sin(m\psi)] \tag{7a}$$

$$\Delta\Phi^A(\eta=0) = \sum_{m=0}^{\infty}\sum_{n=m+1,m+3,\cdots}^{\infty}\bar{P}_n^m(|\nu|)[(\tau_n^{mc})^A\cos(m\psi)+(\tau_n^{ms})^A\sin(m\psi)] \tag{7b}$$

Integration of equation (4) along the free stream direction results in

$$q_i = -\frac{1}{V_\infty}\int_\xi^\infty \Phi^V_{,i}\,d\xi \tag{8}$$

On the other hand, from equation (5),

$$\frac{dq_i}{dt} = -\Phi^A_{,i} \tag{9}$$

Therefore, the z-component of the non-dimensional induced velocity at the rotor disk ($\xi = 0$) is

$$w = -\frac{1}{V_m}\int_0^\infty \frac{\partial\Phi^V}{\partial z}\,d\xi \tag{10}$$

where the integration is carried out along the stream line passing through that point, and its non-dimensional time derivative is

$$\frac{dw}{dt} = \left.\frac{\partial\Phi^A}{\partial z}\right|_{\eta=0} \tag{11}$$

Note that both w and its time derivative are continuous across the rotor disk.

Equations (10) and (11) can be thought of as linear operators,

$$w = L[\Phi^V] \tag{12}$$

$$\frac{dw}{dt} = E[\Phi^A] \tag{13}$$

Assume that the operators $\boldsymbol{L}$ and $\boldsymbol{E}$ are invertable. Then, the time domain induced flow theory (based on

the acceleration potential) can be written in the following form:

$$E^{-1}[\dot{w}]+L^{-1}[w]=\Phi^{A}+\Phi^{V}=\Phi \tag{14}$$

Both operators L and E can be expressed in a matrix form, and can be derived if a proper series of expansion for the induced flow is chosen.

In the generalized dynamic wake theory, the induced inflow distribution at the rotor disk is expanded in terms of a set of both azimuthal harmonics and radial shape functions with underdetermined coefficients as

$$w=\sum_{r}\sum_{j}P_j^r(v)[\alpha_j^{rc}(t)\cos(r\psi)+\alpha_j^{rs}(t)\sin(r\psi)] \tag{15}$$

With this approach, rotor pressure coefficients $\{\tau\}$ and the inflow coefficients $\{\alpha\}$ can be related using the following matrix form

$$[M]\{\dot{\alpha}\}+V_m[L]^{-1}\{\alpha\}=\frac{1}{2}\{\tau\} \tag{16}$$

where, the $[M]$ matrix is called the apparent mass matrix, V_m is the mass flow parameter, and the $[L]$ matrix is the induced inflow influence coefficient matrix. With the $[L]$ matrix and mass flow parameter, one can obtain the quasi-steady part of the induced inflow at the rotor disk for any given load distribution, i.e.,

$$\{\alpha\}=\frac{1}{V_m}[L]\left\{\frac{\tau}{2}\right\} \tag{17}$$

Closed form expressions for the elements of the $[L]$ matrix were developed for out of ground effect flight conditions, and comparisons were made with the theoretical predictions and measured inflow data[9].

Formulation of the Ground Effect Model

In our approach to modeling of ground effect, a second pressure perturbation, Φ_G, is assumed to exist in the flow field due to the presence of the ground plane. We also assume that the acceleration part of this ground pressure perturbation is negligible, i.e., $\Phi_G^A=0$ and $\Phi_G=\Phi_G^V$. The total pressure perturbation is then the superposition of the two contributions from the rotor and the ground, i.e.,

$$\Phi=\Phi_R+\Phi_G=(\Phi_R^V+\Phi_R^A)+\Phi_G \tag{18}$$

For satisfying the Laplace equation and the boundary condition at infinity, the pressure perturbation due to the ground is represented as

$$\Phi_G^V=\frac{1}{2}\sum_{l=0}^{\infty}\sum_{k\geq l}^{\infty}\overline{P}_k^l(\hat{v})\overline{Q}_k^l(i\hat{\eta})[\sigma_k^{lc}\cos(l\hat{\psi})+\sigma_k^{ls}\sin(l\hat{\psi})] \tag{19}$$

where, the associated Legendre functions are now expressed in another ellipsoidal coordinate system $(\hat{v},\hat{\eta},\hat{\psi})$ with its origin at the center of the rotor wake footprint on the ground plane, as shown in Figs. 5 and 6 for hover and forward flight cases, respectively. The transformations between the rotor and the ground ellipsoidal coordinate system, i.e.,

$$(\hat{v},\hat{\eta},\hat{\psi})=F(v,\eta,\psi) \tag{20}$$

and,

$$(v,\eta,\psi)=F^{-1}(\hat{v},\hat{\eta},\hat{\psi}) \tag{21}$$

are determined by flight condition as well as relevant parameters such as the normalized height of rotor above the ground, h, the ground inclination angle, δ, and the wake skew angle, χ.

Using equation (20), the pressure perturbation can be represented as either a source ($k+l$ even) or a pressure-jump ($k+l$ odd) at the ground, each simulating the momentum change due to the turning of the flow by the ground plane. However, the source-like distribution

$$\Phi_G=\frac{1}{2}\sum_{l=0}^{\infty}\sum_{k=l,l+2,\ldots}^{\infty}\overline{P}_k^l(\hat{v})\overline{Q}_k^l(i\hat{\eta})[\sigma_k^{lc}\cos(l\hat{\psi})+\sigma_k^{ls}\sin(l\hat{\psi})] \tag{22}$$

is more reasonable since the flow is turned without energy loss. The axial distributions of rotor and ground pressure perturbations are illustrated in Fig. 7 for h=1.0. The rotor pressure has jump across the rotor disk, while the ground pressure is continuous at the ground surface.

At the ground surface, the ground pressure is

$$\Phi_G=\frac{1}{2}\sum_{l=0}^{\infty}\sum_{k=l,l+2,\ldots}^{\infty}\overline{P}_k^l(\hat{v})[\sigma_k^{lc}\cos(l\hat{\psi})+\sigma_k^{ls}\sin(l\hat{\psi})] \tag{23}$$

and the rotor pressure is

$$\Phi_R=\frac{1}{2}\sum_{m=0}^{\infty}\sum_{n=m+1,m+3,\ldots}^{\infty}\overline{P}_n^m(v)\overline{Q}_n^m(i\eta)[\tau_n^{mc}\cos(m\psi)+\tau_n^{ms}\sin(m\psi)] \tag{24}$$

where (v,η,ψ) in Eq. (24) are related to $\hat{v}$ and $\hat{\psi}$ in Eq. (23) through the transformation

$$(v,\eta,\psi)=F^{-1}(\hat{v},\hat{\eta}=0,\hat{\psi}) \tag{25}$$

By applying the pressure condition

$$\Phi_G=\Phi_R \tag{26}$$

at the ground surface $(\hat{\eta}=0)$ and using the orthogonal property of $\overline{P}_k^l(\hat{v})$ over the interval $\hat{v}\in[0,1]$, i.e.,

$$\int_0^1\overline{P}_i^l(\hat{v})\overline{P}_j^l(\hat{v})d\hat{v}=\delta_{ij} \qquad i,j=r,r+2,\ldots \tag{27}$$

the ground pressure coefficients can be determined as

$$\{\sigma_k^l\}=[B]\{\tau_n^m\} \qquad l,m=0,1,2,\ldots$$
$$k=l,l+2,l+4,\ldots;\ n=m+1,m+3,\ldots \tag{28}$$

where, the elements of the $[B]$ matrix are

$$(B_{kn}^{0m})^{cc}=\frac{1}{2\pi}\int_0^{2\pi}\int_0^1\overline{P}_k^l(\hat{v})\overline{P}_n^m(v)\overline{Q}_n^m(i\eta)\cos(m\psi)d\hat{v}d\hat{\psi}$$

$$(B_{kn}^{0m})^{cs}=\frac{1}{2\pi}\int_0^{2\pi}\int_0^1\overline{P}_k^l(\hat{v})\overline{P}_n^m(v)\overline{Q}_n^m(i\eta)\sin(m\psi)d\hat{v}d\hat{\psi}$$

for $l=0$ (29a)

$$(B_{kn}^{lm})^{cc} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_k^l(\hat{v})\cos(l\hat{\psi})\overline{P}_n^m(v)\overline{Q}_n^m(i\eta)\cos(m\psi)d\hat{v}d\hat{\psi}$$

$$(B_{kn}^{lm})^{cs} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_k^l(\hat{v})\cos(l\hat{\psi})\overline{P}_n^m(v)\overline{Q}_n^m(i\eta)\sin(m\psi)d\hat{v}d\hat{\psi}$$

$$(B_{kn}^{lm})^{sc} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_k^l(\hat{v})\sin(l\hat{\psi})\overline{P}_n^m(v)\overline{Q}_n^m(i\eta)\cos(m\psi)d\hat{v}d\hat{\psi}$$

$$(B_{kn}^{lm})^{ss} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_k^l(\hat{v})\sin(l\hat{\psi})\overline{P}_n^m(v)\overline{Q}_n^m(i\eta)\sin(m\psi)d\hat{v}d\hat{\psi}$$

for $l > 0$ (29b)

The ground effect at the rotor disk can be seen as an interference velocity, w_G, which is defined as positive upward. Thus, the in-ground-effect inflow velocity becomes

$$w^{IGE} = w - w_G \tag{30}$$

where w is the out-of-ground-effect induced velocity. Analogous to the rotor induced velocity, the ground interference velocity can be expressed as

$$w_G = \sum_r \sum_{j=r+1,r+3,\ldots} \overline{P}_j^r(v)[\beta_j^{rc}\cos(r\psi) + \beta_j^{rs}\sin(r\psi)] \tag{31}$$

and the ground interference velocity coefficients β's can be related with the ground pressure coefficients σ's as

$$\{\beta_j^r\} = \frac{1}{V_m}[A]\left\{\frac{\sigma_k^l}{2}\right\} \qquad l, r = 0, 1, 2,\ldots$$

$$k = l, l+2, l+4,\ldots; j = r+1, r+3,\ldots \tag{32}$$

The mass flow parameter in the above equation is the same as that for the out-of-ground-effect case, since the ground does not dissipate the mass flow but only redirects the flow. Using equation (10) and applying the orthogonal property of $\overline{P}_j^r(v)$ over the interval $v \in [0,1]$, i.e.,

$$\int_0^1 \overline{P}_i^r(v)\overline{P}_j^r(v)dv = \delta_{ij} \qquad i,\ j = r+1,\ r+3,\ldots \tag{33}$$

the elements of the $[A]$ matrix can be determined as

$$(A_{jk}^{0l})^{cc} = \frac{1}{2\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^0(v)\int_0^{\infty}\frac{\partial}{\partial z}[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\cos(l\hat{\psi})]d\xi dvd\psi$$

$$(A_{jk}^{0l})^{cs} = \frac{1}{2\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^0(v)\int_0^{\infty}\frac{\partial}{\partial z}[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\sin(l\hat{\psi})]d\xi dvd\psi$$

for $r = 0$ (34a)

$$(A_{jk}^{rl})^{cc} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\cos(r\psi)\int_0^{\infty}\frac{\partial}{\partial z}[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\cos(l\hat{\psi})]d\xi dvd\psi$$

$$(A_{jk}^{rl})^{cs} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\cos(r\psi)\int_0^{\infty}\frac{\partial}{\partial z}[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\sin(l\hat{\psi})]d\xi dvd\psi$$

$$(A_{jk}^{rl})^{sc} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\sin(r\psi)\int_0^{\infty}\frac{\partial}{\partial z}[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\cos(l\hat{\psi})]d\xi dvd\psi$$

$$(A_{jk}^{rl})^{ss} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\sin(r\psi)\int_0^{\infty}\frac{\partial}{\partial z}[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\sin(l\hat{\psi})]d\xi dvd\psi$$

for $r > 0$ (34b)

The coordinate transformation

$$(\hat{v}, \hat{\eta}, \hat{\psi}) = F(v, \eta = 0, \psi)$$

is used to obtain the above integrals.

Combining equations (28) and (32), we have the relationship between the ground interference velocity coefficients and the rotor pressure coefficients

$$\{\beta_j^r\} = \frac{1}{V_m}[G]\left\{\frac{\tau_n^m}{2}\right\} \qquad r, m = 0, 1, 2,\ldots$$

$$n = m+1, m+3,\ldots, j = r+1, r+3,\ldots \tag{35}$$

where the ground effect matrix $[G]$ is defined as

$$[G] = [A][B] \tag{36}$$

or

$$\begin{bmatrix}[G_{jn}^{rm}]^{cc} & [G_{jn}^{rm}]^{cs}\\ [G_{jn}^{rm}]^{sc} & [G_{jn}^{rm}]^{ss}\end{bmatrix} = \begin{bmatrix}[A_{jk}^{rl}]^{cc} & [A_{jk}^{rl}]^{cs}\\ [A_{jk}^{rl}]^{sc} & [A_{jk}^{rl}]^{ss}\end{bmatrix}\begin{bmatrix}[B_{kn}^{lm}]^{cc} & [B_{kn}^{lm}]^{cs}\\ [B_{kn}^{lm}]^{sc} & [B_{kn}^{lm}]^{ss}\end{bmatrix} \tag{36a}$$

Now, the in-ground-effect inflow coefficients are

$$\{\alpha_j^r\}^{IGE} = \{\alpha_j^r\} - \{\beta_j^r\} \tag{37}$$

where $\{\alpha\}$ and $\{\beta\}$ are determined from equations (16) and (35), respectively, for a given rotor load distribution. For quasi-steady condition, the model becomes

$$\{\alpha_j^r\}^{IGE} = \frac{1}{V_m}([L]-[G])\left\{\frac{\tau_n^m}{2}\right\} \tag{38}$$

Finally, the distribution of in-ground-effect induced velocity at the rotor disk is obtained as

$$w^{IGE} = \sum_r \sum_{j=r+1,r+3,\ldots} \overline{P}_j^r(v)[(\alpha_j^{rc})^{IGE}\cos(r\psi) + (\alpha_j^{rs})^{IGE}\sin(r\psi)] \tag{39}$$

As we have seen, not only the magnitude but also the distribution of the ground effect on the rotor inflow are predicted using this model. At this point, two general observations can be made regarding this model. First, the ground interference velocity is computed independently of rotor induced velocity, therefore, no iteration is needed in the solution procedure. Second, since the new in-ground-effect inflow model is still in terms of a set of ordinary differential equations, it can easily be integrated into a flight simulation program.

Hover in Ground Effect

In the hover case, the free-stream can be assumed to be along the rotor z-axis, i.e.,

$$\frac{\partial}{\partial z} = \frac{\partial}{\partial \xi} \tag{40}$$

This fact significantly simplifies the out-of-ground-effect model, resulting in

$$[L] = [I] \tag{41}$$

where $[I]$ is an identity matrix[9]. For the same reason, the $[A]$ matrix is also simplified by working out the integration along the free-stream line,

$$(A_{jk}^{0l})^{cc} = \frac{1}{2\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^0(v)[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\cos(l\hat{\psi})]dvd\psi$$

$$(A_{jk}^{0l})^{cs} = \frac{1}{2\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^0(v)[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\sin(l\hat{\psi})]dvd\psi$$

for $r = 0$ (42a)

$$(A_{jk}^{rl})^{cc} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\cos(r\psi)[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\cos(l\hat{\psi})]dvd\psi$$

$$(A_{jk}^{rl})^{cs} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\cos(r\psi)[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\sin(l\hat{\psi})]dvd\psi$$

$$(A_{jk}^{rl})^{sc} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\sin(r\psi)[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\cos(l\hat{\psi})]dvd\psi$$

$$(A_{jk}^{rl})^{ss} = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \overline{P}_j^r(v)\sin(r\psi)[P_k^l(\hat{v})Q_k^l(i\hat{\eta})\sin(l\hat{\psi})]dvd\psi$$

for $r > 0$ (42b)

For the hover case, the relationship between rotor and ground coordinate systems is shown in Fig. 6. The transformation between the Cartesian coordinate systems is simply

$$\begin{aligned}\hat{x} &= x\\ \hat{y} &= y\\ \hat{z} &= z - h\end{aligned} \tag{40}$$

The transformations F and F^{-1} in equation (20) can be obtained by combining the above relation with the transformation between Cartesian and ellipsoidal coordinate systems (See Appendix). Note that this transformation results in

$$\hat{\psi} = \psi \tag{41}$$

This relation eliminates all the sine-cosine-coupling terms in $[A]$ and $[B]$ matrices, so that the $[G]$ matrix becomes

$$\begin{bmatrix}[G_{jn}^m]^{cc} & 0\\ 0 & [G_{jn}^m]^{ss}\end{bmatrix} = \begin{bmatrix}[A_{jk}^{rl}]^{cc}[B_{kn}^{lm}]^{cc} & 0\\ 0 & [A_{jk}^{rl}]^{ss}[B_{kn}^{lm}]^{ss}\end{bmatrix} \tag{42}$$

In the computation of the ground influence coefficient matrix $[G]$ in this paper, the following set of pressure and inflow coefficients are used:

$$\{\tau\} = \{\tau_1^{0c}, \tau_3^{0c}; \tau_2^{1c}, \tau_4^{1c}; \tau_2^{1s}, \tau_4^{1s}\}^T$$

$$\{\sigma\} = \{\sigma_0^{0c}, \sigma_2^{0c}, \sigma_4^{0c}; \sigma_1^{1c}, \sigma_3^{1c}, \sigma_5^{1c}; \sigma_1^{1s}, \sigma_3^{1s}, \sigma_5^{1s}\}^T$$

$$\{\alpha\} = \{\alpha_1^{0c}, \alpha_3^{0c}; \alpha_2^{1c}, \alpha_4^{1c}; \alpha_2^{1s}, \alpha_4^{1s}\}^T$$

$$\{\beta\} = \{\beta_1^{0c}, \beta_3^{0c}; \beta_2^{1c}, \beta_4^{1c}; \beta_2^{1s}, \beta_4^{1s}\}^T$$

Then, equation (36) for this case becomes

$$[G]_{6\times 6} = [A]_{6\times 9}[B]_{9\times 6}$$

The structure of the $[G]$ matrix for the assumed number of pressure and inflow coefficients is shown in Table 1. As examples, the computed $[G]$ matrices for the cases of hovering at $h = 0.6$ and $h = 1.5$ are shown in Tables 2 and 3, respectively.

Table 1. Example of $[G]$ matrix structure.

$$\begin{bmatrix}
(G_{11}^{00})^{cc} & (G_{13}^{00})^{cc} & (G_{12}^{01})^{cc} & (G_{14}^{01})^{cc} & (G_{12}^{01})^{cs} & (G_{14}^{01})^{cs}\\
(G_{31}^{00})^{cc} & (G_{33}^{00})^{cc} & (G_{32}^{01})^{cc} & (G_{34}^{01})^{cc} & (G_{32}^{01})^{cs} & (G_{34}^{01})^{cs}\\
(G_{21}^{10})^{cc} & (G_{23}^{10})^{cc} & (G_{22}^{11})^{cc} & (G_{24}^{11})^{cc} & (G_{22}^{11})^{cs} & (G_{24}^{11})^{cs}\\
(G_{41}^{10})^{cc} & (G_{43}^{10})^{cc} & (G_{42}^{11})^{cc} & (G_{44}^{11})^{cc} & (G_{42}^{11})^{cs} & (G_{44}^{11})^{cs}\\
(G_{21}^{10})^{sc} & (G_{23}^{10})^{sc} & (G_{22}^{11})^{sc} & (G_{24}^{11})^{sc} & (G_{22}^{11})^{ss} & (G_{24}^{11})^{ss}\\
(G_{41}^{10})^{sc} & (G_{43}^{10})^{sc} & (G_{42}^{11})^{sc} & (G_{44}^{11})^{sc} & (G_{42}^{11})^{ss} & (G_{44}^{11})^{ss}
\end{bmatrix}$$

Table 2. $[G]$ Matrix in normal hover case (h=0.6).

.2245	.0303	.0000	.0000	.0000	.0000
-.0588	.0012	.0000	.0000	.0000	.0000
.0000	.0000	.0483	.0108	.0000	.0000
.0000	.0000	-.0100	-.0005	.0000	.0000
.0000	.0000	.0000	.0000	.0483	.0108
.0000	.0000	.0000	.0000	-.0100	-.0005

Table 3. $[G]$ Matrix in normal hover case (h=1.5).

.0496	.0039	.0000	.0000	.0000	.0000
-.0169	-.0013	.0000	.0000	.0000	.0000
.0000	.0000	.0018	.0002	.0000	.0000
.0000	.0000	-.0006	-.0001	.0000	.0000
.0000	.0000	.0000	.0000	.0018	.0002
.0000	.0000	.0000	.0000	-.0006	-.0001

It can be seen that, in the $[G]$ matrix for the hover case, all of the sine-cosine coupling sub-matrices are zero, and the first harmonic sine and cosine sub-matrices are the same. This is a result of the axially symmetric feature of ground effect in the normal hover case. However, radial coupling does exist in each harmonic sub-matrix, implying that the radial distribution of inflow varies with the rotor height above the ground even for the same load distribution. It also can be seen that the sine and cosine harmonic sub-matrices are an order of magnitude smaller than $[G^{00}]$. Comparing the first diagonal element in each harmonic sub-matrix, the off-diagonal coupling terms are an order of magnitude smaller, and the second diagonal element is even smaller. In other words, the first diagonal element $(G_{11}^{00})^{cc}$ is remarkably larger than the other elements in the $[G]$ matrix. Since the average induced velocity at the rotor disk is determined by the first inflow coefficients[9], i.e.,

$$\overline{w} = 2\alpha_1^{0c}/\sqrt{3} \tag{43}$$

the ground effect on the average inflow is much stronger than that on the azimuthal and radial distributions of the induced velocity.

In the computations in this paper, we assume that the rotor load distribution is kept the same for varying height of the rotor above the ground, and only the 0th harmonic load is considered, i.e.,

$$\{\tau\} = \{\tau_1^{0c}, \tau_3^{0c}; 0, 0; 0, 0\}^T = \text{constant} \tag{44}$$

Two assumed radial load distributions are shown in Fig. 8, where

Load distribution 1 (solid line): $\tau_3^{0c} = 0$

Load distribution 2 (dashed line): $\tau_3^{0c} = -0.6\tau_1^{0c}$

These two kinds of load distribution are comparable to the uniform and triangular distributions, respectively.

The results of average inflow obtained by using the proposed finite-state model for the hover case are shown in Fig. 9, where the average inflow is normalized with respect to the OGE value, and the rotor height above the ground is normalized with respect to rotor radius. The results obtained using Newman's and Hayden's empirical methods are also shown in Fig. 9. Comparing with Fig. 1, it can be seen that the proposed model predicts at least as well as Cheeseman's image method. Especially at low altitude (h<0.4), the proposed model gives better prediction than Cheeseman's method.

Comparing the average inflow corresponding to the different load distributions, it can also be seen that the radial load distribution has an influence on the average inflow for varying height, but the influence is not significant.

The radial distributions of inflow at the rotor disk are computed for varying heights, and the results are shown in Fig. 10. In order to be able to compare with Fig. 4 of Ref. [8], the load distribution is chosen as $\tau_3^{0c} = -0.5\tau_1^{0c}$ as per the trim calculation of the R-50 helicopter rotor (parameters listed in Tab. 4) in OGE hover.

Table 4. Major parameters of R-50 helicopter rotor

Rotor diameter	3.07 m
Blade chord	0.106 m
Gross weight	522 N
Airfoil	DAE 41
Built-in twist	-3.5°
Rotor solidity	0.044
Rotational speed	78.5±1.3 rad/sec
Tip Mach No.	0.36
Re. at 75% radius station	6.50×10^5

It can be seen from a comparison between Figs. 4 and 10 that,

1) the two methods give almost the same prediction on the magnitude of induced velocity around r/R=0.7;
2) the difference between the two predicted radial distributions is mainly because the free wake model neglects the inner vortex sheets, while the finite-state inflow model can't accurately capture the inflow peak induced by the tip vortices;
3) the decrease in induced velocity due to ground effect is also predicted to be at the same magnitude by both methods roughly around r/R=0.7;
4) the proposed model predicts a nearly uniform decrease of inflow over the rotor disk, while that predicted by the free-wake/panel model is more concentrated at the center of the rotor disk.

Forward Flight in Ground Effect

For a helicopter rotor in forward flight close to a ground plane, the relationship between rotor and ground coordinate systems is shown in Fig. 6. The transformation between the rotor and the ground Cartesian coordinate systems can be written in a matrix form as

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{bmatrix} \cos\alpha_{TPP} & 0 & \sin\alpha_{TPP} \\ 0 & 1 & 0 \\ -\sin\alpha_{TPP} & 0 & \cos\alpha_{TPP} \end{bmatrix} \begin{Bmatrix} \hat{x} - h\tan(\alpha_{TPP}+\chi) \\ \hat{y} \\ \hat{z}+h \end{Bmatrix} \tag{45}$$

where α_{TPP} is the angle-of-attack of the rotor tip-path-plane, and χ is the skew angle of the rotor wake.

The transformations F and F^{-1} in equation (20) can be obtained by combining equation (45) with the transformation between Cartesian and ellipsoidal coordinate systems (See Appendix). Note that, as long as $\chi \neq 0$, we have $\hat{\psi} \neq \psi$ even when the rotor disk is parallel to the ground plane. This will bring out the sine-cosine-coupling terms in $[A]$ and $[B]$ matrices as well as in $[G]$ matrix.

The choice of the wake skew angle will have a significant influence on the computation results of $[G]$ matrices. Undoubtedly, it would be best to calculate the actual deformed wake shape rather than to idealize the wake so that it lies along a straight line. However, the complexity and expense of such computation does not appear to be practical for flight simulation study, even if all of the computational convergence problems could be overcome. In practice, it is clear that the use of the skew angle obtained form momentum theory leads to reasonable results for rotor induced velocity[9]. However, Ref. [7] has shown that it is necessary to modify the momentum-theory skew angle to account for roll-up of the rotor wake when calculating the ground interference velocity field. Reference [7] also suggestes that the effective skew angle of rolled-up rotor wake can be determined as a modification of the momentum theory skew angle as

$$\tan\chi_e = \frac{\pi^2}{4}\tan\chi \tag{46}$$

A similar dual skew-angle approach is used here, with the momentum skew angle χ being used to compute the $[L]$ matrix, and an effective skew angle χ_e being used to compute the $[G]$ matrix. Equation (46) is used to calculate the effective skew angle in the present analysis. Examples of computed $[G]$ matrices for the normalized advance ratio μ = 0.5 and 1.0 are shown in Tables 5 and 6, respectively.

Table 5. [G] Matrix for h = 0.68, μ^* = 0.5

.12515	.02372	.02350	.00504	.00000	.00000
-.03909	-.00609	-.00908	-.00163	.00000	.00000
.06887	.01156	.01800	.00430	.00000	.00000
-.01954	-.00234	-.00572	-.00093	.00000	.00000
.00000	.00000	.00000	.00000	.01066	.00158
.00000	.00000	.00000	.00000	-.00351	-.00052

Table 6. [G] Matrix for h = 0.68, μ^* = 1.0

.00824	-.00110	.00402	-.00025	.00000	.00000
-.00376	.00050	-.00184	.00012	.00000	.00000
.00601	-.00082	.00299	-.00019	.00000	.00000
-.00273	.00037	-.00136	.00008	.00000	.00000
.00000	.00000	.00000	.00000	.00011	-.00001
.00000	.00000	.00000	.00000	-.00005	.00000

Compared with the [G] matrix for hover case, the sine and cosine first harmonic sub-matrices of the forward flight [G] matrices are no longer the same in forward flight, implying that the ground plane has different effect on the longitudinal and lateral distributions of the rotor inflow. There are non-zero sub-matrices for coupling between fundamental and cosine harmonics. The coupling sub-matrices associated with sine harmonics are still zeros since the ground effect is symmetric about the x-axis when the bank angle of the rotor tip-path-plane is negligible.

The measurements of Ref. [3] are compared to the results of the proposed model for different rotor load distributions in Fig. 11. Power measurements were not presented for the out-of-ground-effect case in Ref. [3]; thus, the present comparison is referenced to the theoretical in-ground-effect hovering induced power, which is the same as the comparison shown in Fig. 2. The initial increase in induced power as the rotor forward speed is increased from hover is present in both the calculated and measured powers. Comparing Figs. 2 and 11, it can be seen that the results of the proposed model are closer to the measured powers than those of Heyson' image model.

The empirical values of the inflow components from Ref. [5] are used for an additional validation of the proposed finite-state model. In Ref. [5], the inflow at rotor disk is expressed as

$$\lambda = \lambda_0 + \lambda_c \cos\psi + \lambda_s \sin\psi \qquad (47)$$

while the induced velocity w is expressed in the present study by Eq. (15). To make a proper comparison, the inflow components λ's should be written as functions of the inflow coefficients α's. By setting the integral relation as

$$\int_0^{2\pi}\int_0^1 \lambda r dr \cos\psi d\psi = \int_0^{2\pi}\int_0^1 w r dr \cos\psi d\psi \qquad (48)$$

it can be shown that

$$\lambda_c = 2 \sum_{n=2,4,6,\ldots} \alpha_n^{1c} \int_0^1 \overline{P}_n^1(v) v dv \qquad (49)$$

With two radial expansion terms (n = 2, 4), since

$$\overline{P}_2^1 = \sqrt{\frac{1-v^2}{6}}\frac{d\overline{P}_2^0(v)}{dv} = \sqrt{\frac{15}{2}}v\sqrt{1-v^2}$$

$$\overline{P}_4^1 = \sqrt{\frac{1-v^2}{20}}\frac{d\overline{P}_4^0(v)}{dv} = \frac{\sqrt{5}}{4}(21v^2-9)v\sqrt{1-v^2}$$

equation (49) becomes

$$\lambda_c = 2[\alpha_2^{1c}\int_0^1 \overline{P}_2^1(v)vdv + \alpha_4^{1c}\int_0^1 \overline{P}_4^1(v)vdv]$$

$$= \frac{\pi}{64}[4\sqrt{30}\alpha_2^{1c} + 3\sqrt{5}\alpha_4^{1c}] \qquad (50)$$

$$\approx 1.075\alpha_2^{1c} + 0.3293\alpha_4^{1c}$$

Figure 12 shows the cosine (fore and aft) component of the inflow, λ_c, as a function of normalized advance ratio, μ^*, for varying heights normalized to rotor radius, h. For comparison purposes, both the theoretical results and the empirical values are included in Fig. 12. It should be recalled that the values of the rotor height above the ground in the Fig. 3 are normalized to the rotor diameter. The lateral or sine component of the inflow was found to be small and is not presented here. As shown in Fig. 12, the proposed finite-state model is able to predict very well the values of the longitudinal inflow component in low-speed forward flight in ground effect. This effect becomes more significant as the height of the rotor above the ground is decreased. At the lowest normalized height (h=0.46), the proposed model underestimates the inflow gradient at μ^*=0.2 to 0.4, and overestimates the gradient at μ^*=0.6 to 0.8. It is accurate at μ^*=1.0 to 1.2. However, the model successfully shows that the inflow gradient stays fairly constant up to μ^*=0.6 and then abruptly transitions to the out-of-ground-effect value. In summary, the present model is excellent on the average inflow and very good on the inflow gradient.

Conclusions

The major conclusions from this research are summarized as follows:

1) A finite-state in-ground-effect inflow model has been developed for lifting rotors operating close to a ground plane.
2) Ground effect at the rotor disk has the character of an upward interference velocity, which is related to the rotor load distribution by the ground influence coefficient matrix, [G].
3) The elements of the [G] matrix have been determined as functions of the normalized height of the rotor above the ground plane, the tip-path-plane angle-of-attack, and the effective wake skew angle.
4) The structure of the [G] matrix indicates coupling among the radial shape modes and azimuthal harmonics of the rotor inflow, implying that the

ground plane has effect on the inflow distribution at rotor disk.

5) For the in-ground-effect average inflow in hover and forward flight, the correlation between the results of the proposed model and the experimental data is good over all. The new model performs as well as, or better than classical image methods.
6) Results indicate that the induced power of a rotor in ground effect increases, rather than decreases, as the forward speed is increased initially form hover.
7) Results of the proposed model indicate that the fore-and-aft component of the harmonic inflow is reduced by operation close to the ground, which agrees with experimental data.

Acknowledgement

This study is carried out under the NRTC Center of Excellence in Rotorcraft Technology research program at Georgia Tech and Washington University.

References

[1] Hayden, J. S., "The Effect of the Ground on Helicopter Hovering Power Required," Presented on the AHS 32nd Annual Forum, May 1976.

[2] Newman, E. M., "A New Approach to the Calculation of the Effect of the Ground on the Performance of Rotary Wing Aircraft," AD-736925, November 1971.

[3] Huston, R. J. and Morris, C. E. K., Jr., "A Wind-Tunnel Investigation of Helicopter Directional Control in Rearward Flight in Ground Effect," NASA TN D-6118, 1971.

[4] Curtiss, H. C., Jr., Sun, M., Putman, W. F. and Hanker, E. J., Jr., "Rotor Aerodynamics in Ground Effect at Low Advance Ratios," Proceedings of the AHS 37th Annual Forum, May 1981.

[5] Curtiss, H. C., Jr., Erdman, W. and Sun, M., "Ground Effect Aerodynamics," Vertica, Vol. 11, No. 1/2, pp. 29-42, 1987.

[6] Cheeseman, I. C. and Bennett, W. E., "The Effect of the Ground Effect on a Helicopter Rotor in Forward Flight," ARC R&M 3021, 1955.

[7] Heyson, H. H., "Theoretical Study of the Effect of Ground Proximity on the Induced Efficiency of Helicopter Rotors," NASA TM X-71951, 1977.

[8] Itoga, N., Iboshi, N., Nagashima T., Prasad, J. V. R., Xin, H., and Peters, D. A., "A New Numerical Method for Predicting Aero-Mechanical Behaviors of a Rotor Hovering at Close Proximity to Inclined Flat Surface," Proceedings of the AHS 55th Annual Forum, Montreal, Quebec, Canada, May 1999.

[9] He, Cheng-Jian, Development and Application of a Generalized Dynamic Wake Theory for Lifting Rotors, Ph.D. Thesis, Georgia Tech., July 1989.

[10] Xin, H., Prasad, J. V. R. and Peters, D. A., "Ground Effect Aerodynamics of Lifting Rotors Hovering Above Inclined Ground Plane," Presented no the AIAA 17th Applied Aerodynamics Conference, Norfolk, VA, June 1999.

Appendix: Ellipsoidal Coordinate System

The ellipsoidal coordinate system (ν , η , ψ) is defined such that

$$x = -\sqrt{1-\nu^2}\sqrt{1+\eta^2}\cos\psi$$
$$y = \sqrt{1-\nu^2}\sqrt{1+\eta^2}\sin\psi \qquad (A.1)$$
$$z = -\nu\eta$$

It may be noted that this (ν , η , ψ) coordinate system will cover the entire 3-D space once and only once if we restricted ν , η , and ψ to the ranges

$$-1 \le \nu \le +1$$
$$0 \le \eta \le \infty \qquad (A.2)$$
$$0 \le \psi \le 2\pi$$

Figure A-1 shows the (ν , η , ψ) coordinate system viewed in the x-z plane. The ν = constant surfaces are hyperboloids and the η = constant surfaces are ellipsoids, both families of surfaces being azimuthally symmetric about the z-axis. ψ is the azimuthal angle measured from the negative x-axis, counterclockwise looking along the plus z-axis. $\eta = 0$ represents the two faces of the disk, and ν changes sign as one crosses the disk.

The inverse of the (A.1) is

$$\nu = -\frac{1}{\sqrt{2}}\,sign(z)\sqrt{(1-S)+\sqrt{(1-S)^2+4z^2}}$$
$$\eta = -\frac{z}{\nu} \qquad (A.3)$$
$$\psi = \tan^{-1}\left(\frac{y}{-x}\right)$$

where

$$S^2 = x^2 + y^2 + z^2 \qquad (A.4)$$

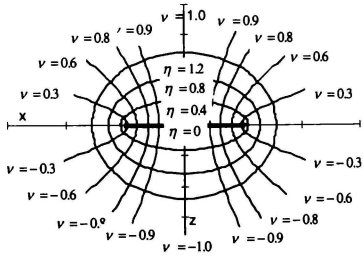


Fig. A-1 Ellipsoidal Coordinate System

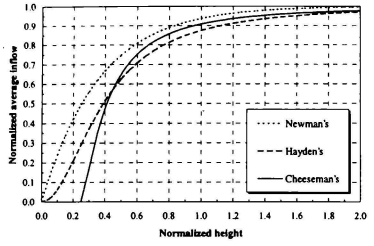


Fig. 1. Empirical[1][2] and analytical[6] results of the in-ground-effect average inflow in hover.

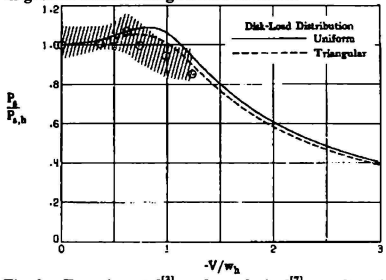


Fig. 2. Experimental[3] and analytical[7] results of the IGE induced power in forward flight. The shaded region shows the range of measured power and the symbols show the mean value. $h = 0.71$.

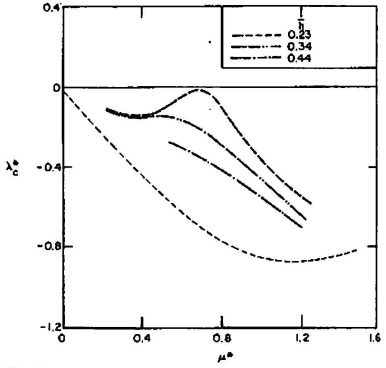


Fig. 3. Cosine component of harmonic inflow in IGE forward flight determined from hub moment measurements[5]. (- - -) Theoretical OGE.

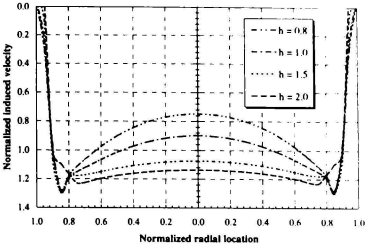


Fig. 4. Radial distribution of induced velocity in IGE Hover from a free wake/panel model[8].

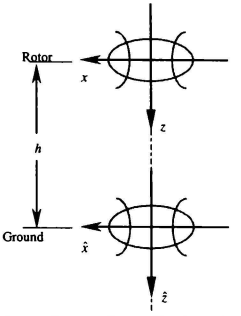


Fig. 5. Rotor and ground coordinate systems.

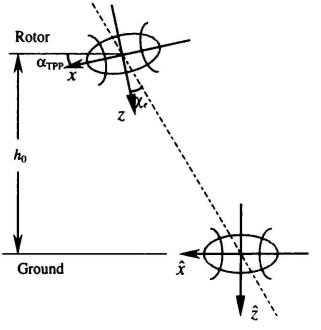


Fig. 6. Coordinate systems for forward flight.

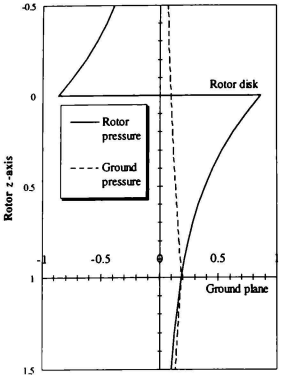


Fig. 7. Axial distribution of rotor and ground pressure perturbations (h=1.0).

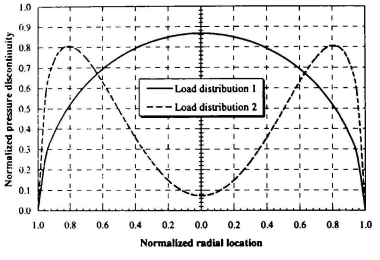


Fig. 8. Assumed radial distributions of rotor load.

Load distribution 1: $\tau_3^{0c} = 0$

Load distribution 2: $\tau_3^{0c} = -0.6\tau_1^{0c}$

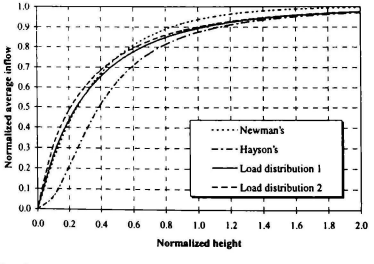


Fig. 9. Comparison between the IGE average inflow predicted by the proposed model and the empirical values from Refs. [1] and [2].

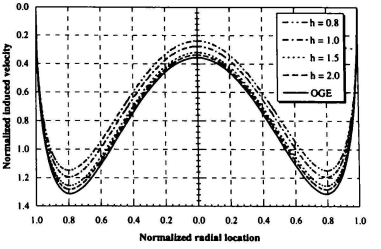


Fig. 10. Radial distribution of inflow at rotor disk predicted using the proposed model.

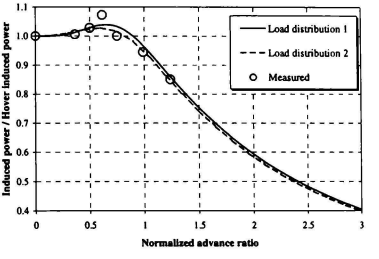


Fig. 11. Comparison between the IGE induced power predicted by the proposed model and the measured data from Ref. [3], h = 0.71.

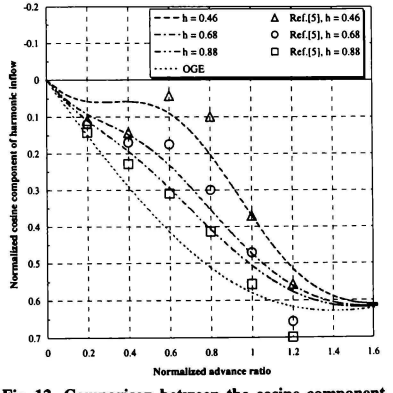


Fig. 12. Comparison between the cosine component of harmonic inflow predicted by the proposed model and the empirical data form Ref .[5].

VIRTUAL BATTLE-SPACE MANAGEMENT SYSTEM: A REAL-TIME SIMULATION DEVELOPMENT AND MANAGEMENT TOOL

Robert C. Subr, 1Lt. USAF
Air Force Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433

Abstract

With simulation environments becoming more complex, it is important to have flexible tools to help visualize the simulation. The Virtual Battle-Space Management System (VBMS) is one such tool. The VBMS software represents a landmark advancement in the generation of high-performance, state-of-the-art virtual/synthetic tools. It was designed to aid in the development, operation, and analysis of real-time mission level simulations. By using several unique viewing methods, VBMS can be used for validation, monitoring, out-the-window visualization for pilot combat stations, subject debriefing, and analysis. Several advanced programming interfaces (API) provide users with a flexible and cost effective tool for research, acquisition, test and evaluation, and training tasks. The C++ design is built around the Silicon Graphics Performer API, which adds a flexible cost effective graphics library for fast prototyping with optimized performance and allows VBMS to operate on the full line of Silicon Graphics computers. This paper will discuss the development, operation, and design of the VBMS Software.

Background

VBMS was designed at the Air Force Research Laboratory, Integration and Assessment Branch (AFRL/VACD). AFRL/VACD is dedicated to the advancement and transfer of technology for military applications. The branch uses ground-based simulations for integrating and assessing current and future aerospace technologies.

As an in-house tool, VBMS allows engineers to develop and evaluate simulation interactions throughout the entire simulation process. The original VBMS design allowed engineers to monitor simulation operations from a third person perspective. This 3D view enabled the user to move about the synthetic environment independent of or attached to an entity.

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

During a simulation study, it was noticed that VBMS could be modified to fit requirements for a unit training device (UTD) tactical display. Requirements were generated to fill the need for the tactical display in a instructor operating station (IOS) for the UTD. These requirements called for a low cost monitoring tool to be integrated into the UTD using a low end Silicon Graphics (SG) Computers. It was decided that the basic viewer functionality would be extended to include overhead maps and a pilot out-the-window (OTW) view. The overhead maps or plan view maps (PVM) allowed the user to zoom between multi-layered pilot charts. These charts are products from the National Imagery and Mapping Agency (NIMA). The OTW view would allow an instructor using a simple flight stick at the console, to fly the simulation or see what the pilot is seeing. In addition, the original functionality of the 3D observer was enhanced to include post analysis flight tapes and other useful markings. Modifications were also made to enable the software to interact with the simulation to setup initial conditions. However at that time only fifty entities could be monitored at once. All communication with the simulation was done through Distributed Interactive Simulation (DIS) version 2.0.4. These requirements are the foundation of the VBMS of today.

Shortly following the IOS version, VBMS was chosen by Aeronautical Systems Center's new Simulation and Analysis Facility (SIMAF) for use during a demonstration of network connectivity to the Theater Air Command and Control Simulation Facility (TACCSF) for the Warrior Flag 97 exercise. During Warrior Flag 97, VBMS was used to monitor both synthetic and live forces over several days. Because of the large number of entities participating in this exercise, VBMS was enhanced to support larger amounts of data. Other options such as filtering and event menus were added to enable the users to locate and monitor specific entities. These menus and monitoring tools were the first analyst functions added to VBMS. In addition, Playback and Record were also added to enable the simulation to be replayed without the need for a DIS logger. These changes were major and furthered the design.

Following the Warrior Flag development, changes were made through other simulation project requirements. One program in specific made a major contribution to the development of the VBMS software. In 1998 AFRL/VACD began work on a larger virtual environment. For these tests, VBMS was used to validate simulation systems, ranging from entity movement to threat emissions and tracking.

During simulation operation, VBMS displayed a complete picture of the mission battlefield and the interactions of well over a thousand entities simultaneously. Throughout the tests, the simulation was recorded by VBMS. After the testing, playback and other utilities in VBMS were used to capture a complete mission timeline and interactions. The playbacks of the entire mission aided in the analysis and debrief of the pilots. Using this tool, pilots could see a clear visualization of their missions during debrief. This also allowed for faster analysis and reconstruction of mission information. Analysts were able to make conclusions about the testing in a matter of days versus weeks.

These tasks were made possible by major improvements made during the simulation development. Architectural changes were made to allow VBMS to process even more entities, and more information than before. Additionally, many new types of monitoring viewpoints and tools were added. At the same time as this development, several contractors and government organizations were also using and modifying VBMS to meet their own needs. One contractor reworked the Graphic User Interface (GUI) for VBMS and integrated it as the test director for an air to air performance analysis simulation. These changes improved the user interface and added a rich set of operation controls. Further changes provided the ability to spawn and manage up to six individual viewports, and added the ability to limit display complexity for better performance on lower end computers. Other enhancements included the use of Standard Template and Rogue Wave libraries to simplify code design and aid in future developments. Contributions like this ensure that VBMS will continue to grow and enhance simulation environments in the future.

Virtual Battle-space Management System

VBMS can be devided into two sections, the control interface or GUI and the resizable graphic viewport displays. Figure 1 is an example of VBMS during execution. Any of the graphics viewports or control interface can be resized or repositioned on the computer screen.

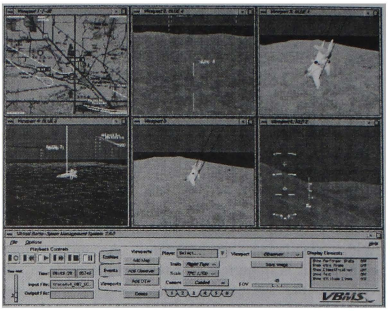


Figure 1 - VBMS 2.0

VBMS Graphic Viewports

VBMS supports six active resizable viewports. Each viewport can operate in one of three modes: observer (Figure 2), plan view maps (Figure 3), and out the window (Figure 4).

The observer mode (Figure 2) for simulation monitoring allows users to detach from any existing entity or object and move around freely in the database.



Figure 2 - Observer View

The observer view allows for several types of markings, trailing lines, and sensor information. Different configurations can be set through the control interface to display the most important information. Additionally, various track modes can be enabled for better monitoring. Users can attach themselves to active objects and follow in trail, cabled, or tracking

from stationary position. Observer views are excellent for runtime interaction analysis and system operation testing.

The next viewport type is the PVM. The PVMs of the battlefield give a grand perspective of the area of interest. PVMs provide a simple global picture of a complex simulation. Utilizing pilot charts and satellite imagery provided by National Imagery and Mapping Agency, the VBMS PVMs show a representation of entity movement, sensor interactions, weapon tracks, emission tracks, event information, and kill markings.

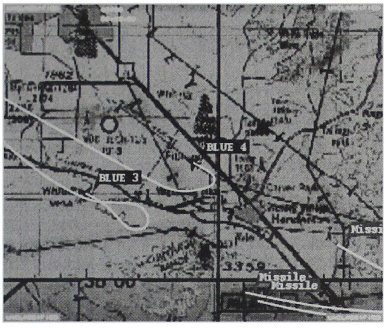


Figure 3 - Plan View Map

Figure 3 shows several entities in the PVM. The user can select an entity to center on and track, or just pan freely around the database. Range and bearing information between entities or to points on the ground is also available. PVMs also provide the ability to zoom between different resolutions ranging from a global perspective down to 5-meter satellite imagery.

In addition to the PVMs and Observer views, VBMS can attach to an entity and function as an out the window view (Figure 4). Like the observer view, the OTW view provides a 3D perspective of the battle-space. Objects can be selected through the control interface. Users have the ability to slew the eye point from the OTW display to mimic a pilot looking around the cockpit. Display information from a simulation such as using a helmet mounted tracking device can also drive this eye point.

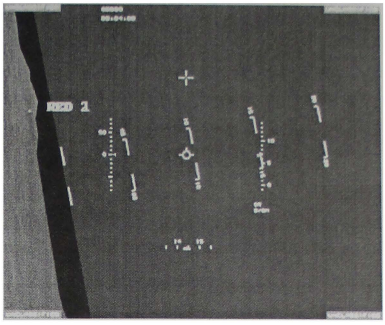


Figure 4 - Out The Window

Display markings such as trailing lines, altitude markings, and text identifiers can also be displayed in this mode.

Using a combination of the three VBMS view types, VBMS can be a very powerful tool. Some typical uses are simulation diagnostics, simulation validation, simulation monitoring, simulation operator console, pilot debrief and simulation analysis. What makes all this possible is the flexible design model. Additionally, if a desired configuration or markings are not available, software changes can be made to the well structured code.

VBMS Control Interface

The VBMS control interface GUI (Figure 5) manages all playback, viewport, entity monitoring, event recording and system options. This GUI is made up of four sections: the playback control, event log, entity listing, and viewport controls.



Figure 5 - VBMS Control Interface

The playback control (Figure 6) acts as the VCR for VBMS. This functionality allows recording of the simulation, and real-time playback.

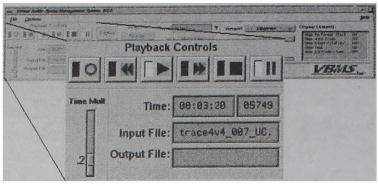


Figure 6 - Playback Controls

The playback control shows simulation time, frame count, a playback or input file, and the output or record file. There is also a time multiplier bar to adjust playback speed between .2 times and 3.5 times real-time.

Another feature of the GUI is the event log (Figure 7). This feature is selected from the event vertical tab and will appear on the right side of the control interface. This log lists simulation events such as emission changes, weapon detonation, entity destruction, and time encoding. This log can be saved to disk for future analysis.

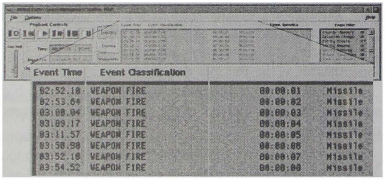


Figure 7 - Event Log

If desired, an event filter can be activated to mask out undesired event listings such as entity births, emission changes, and weapons fire. This helps declutter large amounts of information.

On the same vertical bar is the entity listing tab (Figure 8). This tab displays the entity information listings. These listings give a constant readout of entity data.

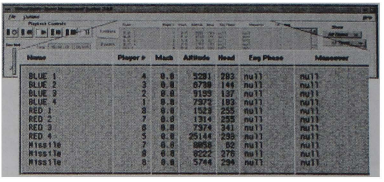


Figure 8 - Entity Listing

Information such as object heading, speed, and current actions can be shown on this table. For simulations with a large amount of entities, the entity filter can be set to limit the type or number of entities shown on this table.

The final tab on the vertical tab bar is the viewport control tab (Figure 9). When selected, this tab will show the options for one of the six possible viewports. From this window, viewport options such as the selected entity, information markers, simulation statistics, and viewport modes can be changed. New viewports can be added using this window. When added, all viewports will automatically be resized to make room for the new viewport. A tab will be added to the control window for the new viewport.

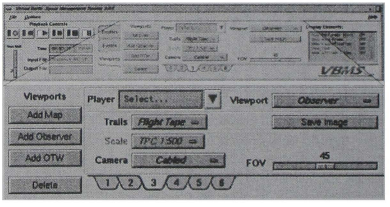


Figure 9 - Viewport Controls

Other functionality of this tab consists of controlling options for the three basic graphics view. As stated above these view types are the PVM, OTW, and observer views. By changing the view type, specific options for that view will be displayed on the control interface. The control interface can manage up to six independent viewports at once. Each viewport can display any of these three graphic configurations and swap configurations at any time during operation.

VBMS Process Design

The VBMS framework is designed around the Silicon Graphics Performer API. Performer uses a process

design that allows the user to optimize the application using a multi-stage application. This multi-stage design allows for better application optimization and various process configurations. The following figure gives an example of a simple Performer process (figure 10).

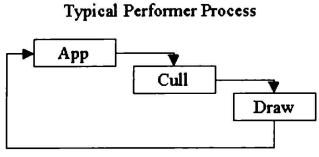


Figure 10 - Simple Performer Process

During the App process, objects are moved, user input is processed, viewing information is altered, and playback or record operations take place. All data in a Performer process is stored in a shared arena. This enables Performer processes to share information at runtime. Once all App manipulation is complete, the Cull process begins. All scene data is preprocessed through what Performer calls the Cull process. During the Cull process, the Performer API "culls" or removes data that is not viewed, or is not needed in the scene. When this process is done, the drawing of the data takes place in the Draw process. Using this model, the Performer API allows for many different processing modes. The App, Cull, and Draw processes can run all in one system process or in several different user selectable configurations.

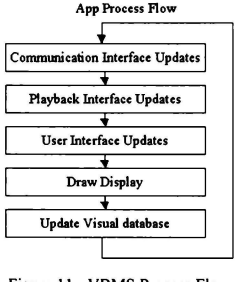


Figure 11 - VBMS Process Flow

The VBMS utilizes the Performer process as the core framework for optimal graphics performance across all levels of SGI computers. Figure 11 shows the order of operations for the VBMS process. The VBMS process flow consists of a modular C++ design. The core modules of VBMS consist of the Graphic User Interface, the shared memory arena, object control module or object manager, database display module, interface control module, and the playback control module. A simplified structure configuration is shown in the following figure (Figure 12).

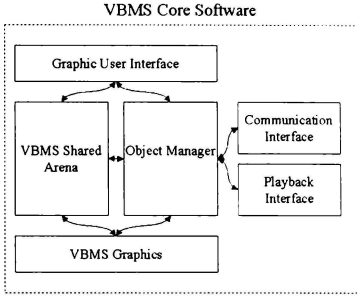


Figure 12 - VBMS Modular Design

Graphic User Interface

All operations of VBMS start with the GUI or control interface. The GUI communicates to the other modules through the shared arena and the object manager. The GUI controls all viewport options, playback controls, entity monitor selection, as well as data collection for runtime or post runtime analysis. Development of this module was done using the Silicon Graphics Rapid-App tool kit. Rapid-App allows for visual construction and modification of the VBMS GUI. Rapid-App generates C++ code, which merges with the current GUI code when changes are made. This allows for an easy upgrade or modification.

Data Storage Arena

The VBMS software maintains all runtime data, and control data using the Performer API shared arenas. These arenas allow VBMS to communicate through several process configurations. By using this predefined scheme, VBMS uses the shared arena to store common control data structures and object definition classes. Other modules such as the object manager access the arena to update entity state information and other parameters.

Object Manager

The object manager controls all entity information interchanges which determines entity state changes, the birth of an entity, the death of an entity, and all entity queries. As entities are received, they are classified and cataloged into categories. A master tree structure is created of categories that are based on team types, vehicle or weapon types, and identification indexing. These categories aid in the speed and operation of the system. By using this type of tree structure, incoming data and information queries for entities are greatly increased. Figure 13 shows a simplified example of the object query in storage space.

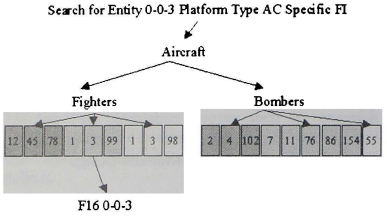


Figure 13 - Data Search Structure

By using specific identification information, a storage tree can be traversed quickly and the object can be identified. A pointer to the object is returned from the search call and allows direct access of the object. Additional entity information is broken down into several types of records. These records consist of sensor, communication, event position, and user defined information. This data can update at different rates. Thus, different groups of data can be presented to the object manager to be altered or recovered. By doing this, only the entity information that the system needs is accessed and this increases the amount of information flow possible within VBMS.

The object manager also retains knowledge of any data or group of data that has changed. It is important for other modules to know when data changes in order to help optimize internal performance. Modules such as the graphics and the playback check for changes when accessing the data. By keeping track of such changes, these modules can optimize their functionality. The object manager also allows for group functions. Group functions allow for operations to be done on specific sets of objects. By using the storage structure, a group or branch of objects can be operated on in one call, without having to look for individual objects in the entire list. An example would be displaying a marker for ground moving objects. Knowing that not all objects are ground objects allows VBMS to access objects on the ground branch of the storage structure and test for changes. Then, if the object is moving, the marker is drawn.

Playback Interface

The playback interface communicates with VBMS through the object manager. Like a video cassette recorder, the playback interface is able to record, play, fast forward, reverse, pause, and stop. In order for the playback to operate with various information at different update rates, playback files are recorded with a record pair. When new data is evaluated, the object manager determines if data has changed and if so, the change is noted. When recording, only information that has changed will be saved to the file. This reduces the amount of data per frame that is recorded. In addition the previous data will also be saved. These recordings are saved as current and previous record pairs. This is done for several reasons. Since only changed data will be saved, the playback interface will not know how far back in the playback file the previous data change was made. It is necessary to know what the previous state of the data is at the current frame in order to reverse the playback. The following figure illustrates the record pairs for one object over time (Figure 14).

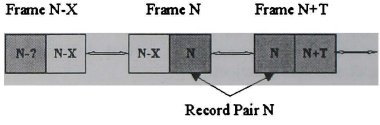


Figure 14 - File Record Pairs

When playing back a recording, the playback interface will only read the current records. For this object, at frame N-X, data block N-X is read, at frame N, data block N is read, and at frame N + T, data block N + T is read, and so on. When the playback interface is reversing a playback, the data blocks switch from current to previous. In rewind at frame N + T, data block N is read, at frame N, data block N – X is read, and at N-X, the previous data block is read. Although this does add some extra overhead, the runtime playback speed is greatly increased and again optimizes the performance of VBMS.

VBMS Graphics Database Design

The VBMS concept was designed to give the user the most information possible without recreating the full up

simulation. Designed around the stealth viewer concept, VBMS incorporates 2D PVMs and 3D synthetic scenes that run at real-time. This module was designed using the Silicon Graphics Performer API and OpenGL 1.1. The Performer API was designed for high detailed visual simulation development. By using Performer as the base graphics API, system development time is reduced. Performer also includes over 30 standard database loaders, which allow the use of many commercially produced databases or models.

The graphics database design for VBMS can be divided into two parts: 2D and 3D visual databases. Although the 2D and 3D databases appear to be separate types of displays, they are linked in position and orientation.

When a user views an overhead map or 3D terrain, they are actually looking at the same area as a 2D map. VBMS is set up to minimize computations and position calculations. Using properties from the draw process, it is possible to link a 2D and 3D database. Figure 15 illustrates this structure.

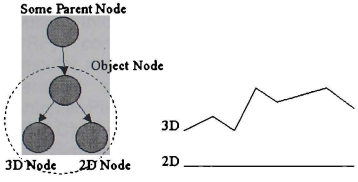


Figure 15 - Database Node

By layering the 3D and 2D maps together, viewing positions and calculations that are done for one map affect both. The figure above shows one database node. This node contains one parent node with two children. Each child represents a piece of the database, an overhead 2D map, and 3D terrain. The databases are of the same area. A draw traversal mask is set on each map type: a traversal mask for 2D and a traversal mask for 3D. A viewport can quickly switch the type of map it is looking at by changing its own traversal mask. If a viewport has selected a 2D mask, the draw process will prune the 3D map from the tree and it will not be drawn during the current frame. Likewise, if a viewport selects a 3D mask, the 2D map will be pruned. This operation takes advantage of Performer traversal mask functionality and helps optimize VBMS operation. This concept also works for the moving objects in the scene. Since icons are 2D and shown on the PVM displays, and 3D models are shown in the OTW and observer displays, this structure shown in Figure 15 is setup for the object models as well.

Simulation Interface

The final major module to be discussed is the communications module. This module communicates with the outside world or simulation. This interface has been designed for a generic shared memory interface. This interface allows users to write their own interface process to communicate with VBMS. The reason for this is to make all interfaces independent of VBMS code. A user can use a sample interface class to write a specific interface to a unique simulation without changing any VBMS code, or if the simulation has the capability, can use commonly defined interfaces such as Distributed Interactive Simulation or High Level Architecture (HLA). The basic design around the simulation interface is the shared memory. Other system processes defined by the user can link to this shared memory. Additionally, this shared memory can be allocated in reflective memory and the interface process may execute on a different computer. The following figure illustrates the interface design (Figure 16).

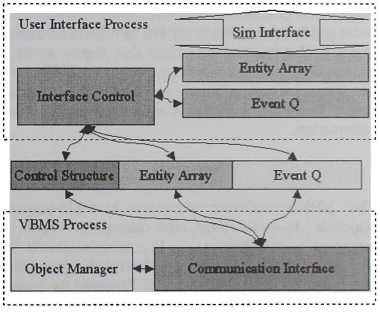


Figure 16 - VBMS Interface

The shared memory structure is defined by a control structure, entity record array, and event queue. Although a process can link directly into the shared memory, to ensure optimal performance, some optimization must be done in the interface process.

Simulation data is received through user defined methods. This data is stored in records and added to an array for basic object definitions. Other information such as events, emissions, and user-specified information are added to an event queue. The interface

control structure communicates through the shared memory control structure to VBMS. This allows VBMS to tell the interface what kind of data it needs. Since the entity array and event queue are of a finite size in the shared memory, data must be paged through the shared memory.

Depending on the types of object records, the interface will typically support up to 512 objects during one frame. This limitation is for several reasons. First, the shared memory size is finite, and secondly, if VBMS is processing an infinite amount of object data per frame, the frame rate will lag or even become deadlocked. Since many simulations deal with more than 512 entities, it is necessary to utilize a paging scheme for VBMS.

During initial execution, VBMS queries the interface for an ordered listing of the current entities. If the current number of objects is greater than 512, then the objects are paged 512 objects at a time until VBMS has a mirror of the interface array. After this initial state, only the objects that are moving, new, or being removed are contained in the object array in shared memory. Hence, at runtime, the simulation will update without paging 512 moving entities in one frame. If more than 512 entities are moving, memory paging will occur. All information is passed to VBMS through a queue. This is done to ensure that events occur in order. The event queue will support up to 2048 events per frame. Event information includes sensor tracks, sensor positions, kill/miss events, and user defined information.

Summary

The VBMS software provides a multi-purpose, scaleable tool to meet the changing needs of tomorrow's simulations. In today's world of simulation, the needs and complexity of a simulation can change from day to day. Through its unique combination of displays, VBMS can be used for simulation diagnostics, validation, monitoring, training, debrief, analysis, and much more. This adaptability is made possible by well-structured designs and API's such as Performer, and Rapid App. Flexibility such as this will enable VBMS to meet the future needs of tomorrow's simulations. Future development is constantly guided by changes and needs of the user community. This ensures a continual and productive growth for VBMS.

ATN APPLICATION DEVELOPMENT AND EVALUATION ON ENRI ATN SIMULATION TESTBED

Kakuichi SHIOMI*, Ken ITANO** and Ryuji OTSUKA***

(*, **) Electronic Navigation Research Institute, Ministry of Transport
6-38-1, Shinkawa, Mitaka, Tokyo 181-0004, JAPAN
(***) OKI Electric Industry Co. Ltd.
4-10-3, Shibaura, Minato-ku, Tokyo 108-8551, JAPAN

Abstract

In 1995, development was started on an Open System Interconnection (OSI) router for the Aeronautical Telecommunication Network (ATN), which will be a vital component of the future air traffic control (ATC) environment. In 1998, OKI Electric Industry Co. Ltd. developed an original Japanese OSI router with Inter-Domain Routing Information Exchange Protocol (IDRP) compatibility on a personal computer running Microsoft's Windows NT operating system. A trial exchanging Controller-Pilot Data Link Communication (CPDLC) messages between the ENRI ATN Simulation Testbed (EAST) and a testbed at Eurocontrol was successfully carried out on 17-18 December 1998.

Currently, ATN applications, such as an aircraft position data exchanger that will also deal with the data from Automatic Dependent Surveillance (ADS), ATC Inter-Facility Data Communication (AIDC), Flight Information Service (FIS), etc., are being developed for the EAST. When designing such ATN applications, it is very important that the delay of data propagation via the ATN be taken into consideration. Empirically, it is known when the response times from the ATC system and/or pilots are greater than several seconds, this can cause stress for the air traffic controllers.
In this paper, details of the EAST and the planning of the ATN application development and evaluation are described.

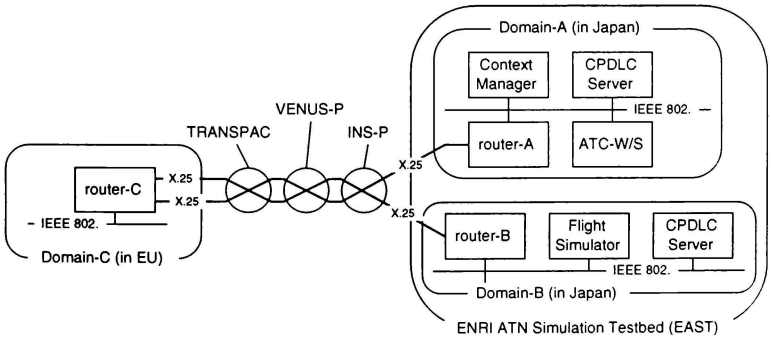


Figure 1. Configuration of Connection Experiment

Introduction

In 1995, ENRI and OKI began development work on an OSI router for an ATN which will be the backbone of future ATC environments. The development of the first Japanese router was completed in 1998, and on 17-18 December of the same year, this router was successfully tested in connection tests conducted jointly with Eurocontrol. The configuration of the system for these mutual connection tests is shown in Fig. 1. The test bed employed in these tests in the Japanese domain is called EAST.

The goals of EAST are to test the ATN draft standard prepared by the ATN panel of the ICAO, to implement the International Connection Experiment, and to conduct research on air traffic controller operations in the ATN environment. Currently in the EAST program, in addition to the router, the Context Management (CM) and CPDLC systems have been fully developed, and development of ADS and AIDC is now under way.

Fig. 2 shows a prototype of an ATC workstation for CPDLC operations. This workstation has features which enable CPDLC messages to be created very easily using a dedicated input unit. Further details of EAST are given below.

EAST

EAST OBJECTIVES

EAST is being developed with the following objectives in order to realize an ATN which will provide the backbone infrastructure for future ATC environments:

a) Test draft standards by development of Router and End Systems (ESs) (such as CPDLC server) based on ICAO draft standards.
b) Verify mutual connectability by the International Connection Experiment, and eliminate problems based on differences in interpretation of ICAO draft standards.
c) Conduct studies on user interfaces with air traffic controllers in order to efficiently utilize ATN applications such as CPDLC.

EAST CONFIGURATION

EAST is made up of groups of equipment that can be categorized according to the following three levels.

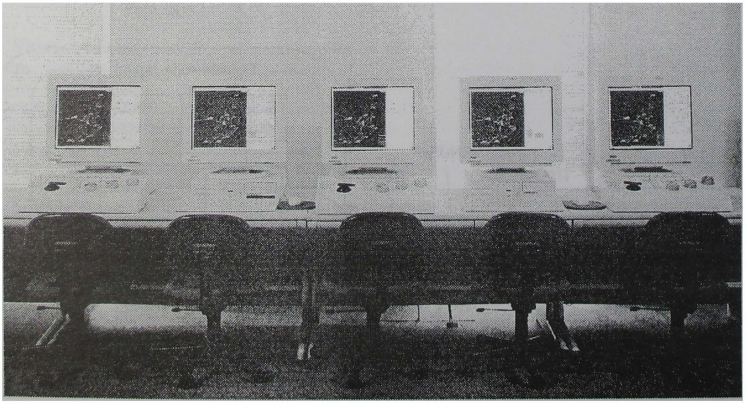


Figure 2. Prototyped ATC Workstation for CPDLC Operation

a) ATN equipment
b) ATC Workstations
c) Simulators

ATN equipment comprises the routers and various ESs used in testing the ICAO draft standards and in the mutual International Connection Experiment.
ATN routers operate on personal computers running the Windows NT operating system. By equipping these computers to handle the Inter-Domain Routing Information Exchange Protocol (IDRP), the ATN routers can also be used for inter-domain communications.
ESs also operate on personal computers running Windows NT. Development of the CM and CPDLC ESs is now complete, and applications are now being developed for AIDC and ADS, in that order.
An example of how a protocol stack is configured in such ATN equipment is shown in Fig. 3.

ATC workstations are being developed for conducting research on air traffic controller user interfaces for efficiently employing ATN applications, and for use as test tools in the International Connection Experiment. The ATC workstation shown in Fig. 2 is a prototype model developed for CPDLC operations. A controller can input CPDLC messages very easily by using a dedicated input unit comprising a tracker ball, three dials, and several buttons. For example, the heading change instruction "Turn left heading 260" can be generated simply by turning the heading dial. Development work on ATC workstations for ADS and AIDC is now being carried out concurrently with ES development.
Simulators provide an environment for conducting mutual International Connection Experiments or evaluating ATC workstations. These simulators comprise the following components.

a) Scenario implementation controller
b) Aircraft simulators
c) Pilot simulators
d) Aircraft systems simulators

The scenario implementation controller dispatches pre-generated flight plan data to the aircraft simulators or ATC workstations and generates a trigger to start the simulation. Based on the flight plan data, the aircraft simulators calculate

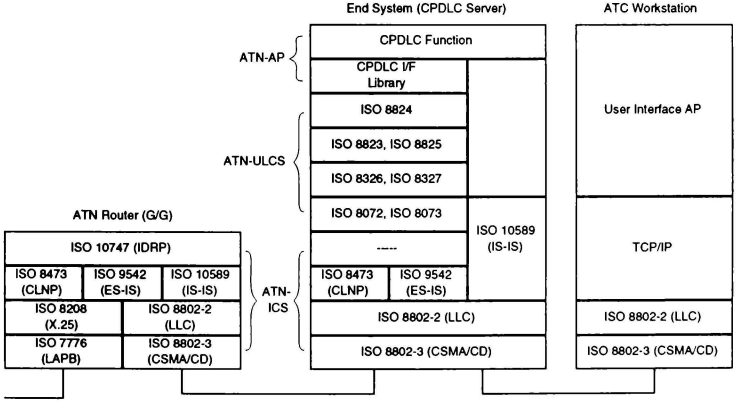


Figure 3. EAST Protocol Stack

aircraft positions in real-time, taking into consideration such factors as aircraft flight characteristics, wind direction, and wind speed. The results are used to generate the displays on the radar screens of the ATC workstations, etc.

In simulation configurations which do not employ aircraft systems simulators, pilot simulators have functions for conducting CPDLC communications with ATC workstations. For example, replies such as "Wilco" or "Unable" are automatically sent in response to controller instructions. When a pilot responds with "Wilco" to a controller's instruction, the aircraft simulator is requested to generate control inputs to comply with that instruction. In response to the instruction "Turn left heading 150," for example, control inputs are generated to cause the aircraft to turn to a heading of 150 degrees.

The aircraft systems simulator simulates the pilot's interface in the cockpit. This simulator can both display and create CPDLC messages, and send and receive CPDLC messages to and from an ATC workstation.

EAST SYSTEM

The configuration of EAST is shown in Fig. 4. The system is configured over multiple domains, each of which contains a router having IDRP functions. A network is formed by interconnecting the routers.

In CPDLC communications, for example, a message is created at an ATC workstation in domain A and sent to a CPDLC server in domain B via the originating CPDLC server, router A in domain A and router B in domain B, according to the CPDLC protocol. The data received by the CPDLC server in domain B are sent to the aircraft systems simulator and displayed on a screen.

Mutual International Connection Experiment

International Connection Experiments using EAST are currently being planned in cooperation with Eurocontrol. These are being conducted in three stages.

(1) Router connections: December, 1998.
(2) CPDLC connections: November, 1999.
(3) ADS connections: September, 2000.

The first stage, the router connection test, was conducted over two days (17-18 December 1998) and verified mutual

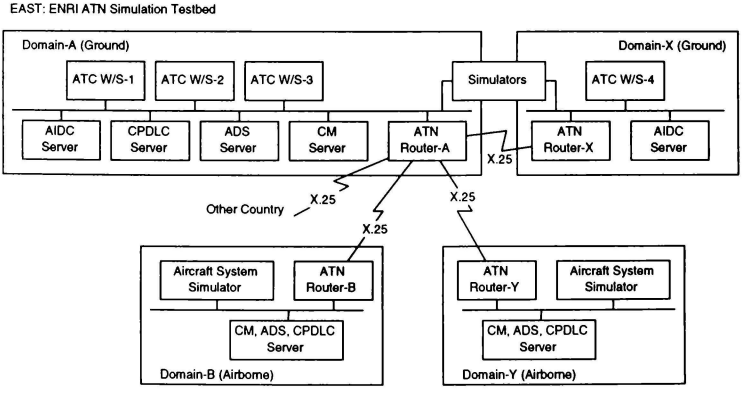


Figure 4. Configuration of EAST System

connectability. More specifically, various Protocol Data Unit (PDU) transmissions and receptions between ATN routers, and the propriety of associated router operations (for example, processing of routing information), were verified.
The configuration of the systems used in these tests is shown in Fig. 1. In this configuration, two routers in Japan and a router in Eurocontrol were connected via INS-P (carrier: NTT), Venus-P (carrier: KDD), and TRANSPAC (EU), and CPDLC data created in domain A in Japan were sent to domain B in Japan after first passing through domain C in Eurocontrol.

In order to conduct mutual connection tests with routers or ESs, it is necessary to coordinate the specifications used at both ends beforehand. This is due to the fact that parties which develop ATN equipment do not necessarily develop all of the standard functions. It is particularly necessary to adequately coordinate specifications for any options that are used. Protocol implementation conformance statements (PICS) were therefore produced in order to ensure specification conformance. PICS are routinely created in the course of developing a router, but there is no standard notation format in ES. PICS for ES are now being prepared on the basis of a Eurocontrol proposal, and this is proving effective in coordinating specifications between the two sides.

ATC Workstation Evaluation Testing

In parallel with the development of ATN equipment, research and development has been conducted on user interfaces for CPDLC operations since 1995. One result of this research is the ATC workstation shown in Fig. 2 (Ref. [1], [2]).

At the end of 1998, evaluation tests were conducted on the ATC workstation for CPDLC operations. These tests were conducted in real-time simulations involving operations by current air traffic controllers. The configuration of the evaluation test system is shown in Fig. 5, and an overview of system operations is shown in Fig. 6. In this configuration, controllers exchange CPDLC messages with a pilot simulator. The evaluation test scenarios employ an average traffic model for terminal control at Tokyo International Airport.

As a result of the tests, most of the controllers found the system acceptable, and the following recommendations were given for further refinement of the user interface.

a) Additional function to enable simultaneous transmission of multiple instructions.
b) Improvement in the layout of the buttons on the input device.
c) Progress display.
d) Study of a user interface taking into consideration the time interval between the transmission of a control instruction and the receipt of the pilot's reply.

It is now planned to modify the ATC workstation in response to these controller reco mmendations. In addition, we plan to develop ATC workstations for efficient ADS

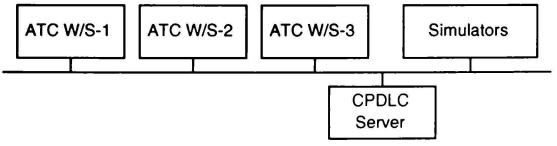


Figure 5. Configuration of the ATC Workstation Evaluation Test System

and AIDC operations, and also to extend our research and development efforts to operation systems integrating CPDLC, ADS, and AIDC.

References:

[1] Otsuka, R. et. al., "Designing of Air Traffic Management System for Terminal Area", AIAA-97-3678, Aug. 1997.
[2] Shiomi, K., et. al., "GUI Design for Tower and Terminal ATC Co-operation", AIAA-98-4290, Aug. 1998.

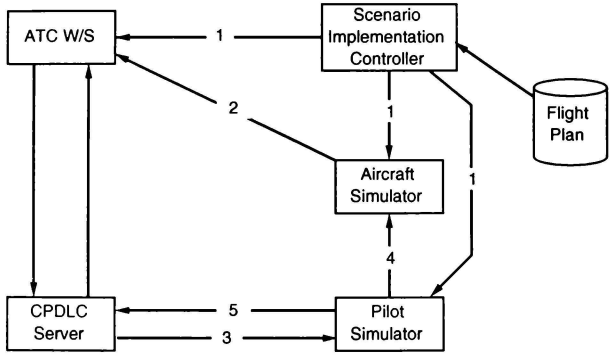


Figure 6. Overview of ATC Workstation Evaluation Test System Operations

1. Dispatch Flight Plan and Commencement Scenario.
2. Sending Aircraft Position Data.
3. Transmission Instruction.
4. Control Aircraft
5. Reply of Pilot's Message.

High Fidelity GPS Satellite Simulator – A Training Tool

Mark Baker, Lockheed Martin Mission Systems – Colorado Springs, CO

Abstract

The need for high fidelity simulation is increasing within the arena of government contracting as the systems that are procured become larger and more complex. The Global Positioning System (GPS) satellite constellation is a national resource that provides an accurate method for precise global navigation. This system, which is operated around the clock by the U.S. Air Force, requires a high degree of accuracy for real-time operations. Lockheed Martin Mission Systems has developed a high fidelity simulator for the Air Force's GPS command and control operations. The simulation of the constellation and its ground control components will enable the Air Force to conduct highly realistic training in a controlled environment to promote operationally-ready satellite controllers.

Introduction

GPS is a spaced-based radio navigation, time distribution and nuclear detonation (NUDET) detection system. Its mission is to provide precise, continuous, all-weather, three-dimensional (3-D) position, velocity, timing, and NUDET information to properly equipped air, land, sea, and space-based users. The system serves various military and civil operations as a highly accurate positioning and navigation data reference. Military operations include, but are not limited to, enroute navigation, low-level navigation, target acquisition, close air support, missile guidance, command and control, all-weather air drop, sensor placement, precise survey, and instrument approach. Civil applications include, but are not limited to, air, land, and sea navigation, surveying, and search and rescue.

Copyright © 1999 by Lockheed Martin. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

As a NUDET detection system (NDS), it detects visible light, X-rays and electromagnetic disturbances emanating from NUDETs. GPS consists of three segments: the Space Segment; the User Segment which consists of the NUDET detection users and the navigation users; and the Control Segment which includes the Operational Control System (OCS). The Operational Control System includes the Master Control System (MCS), multiple Ground Antennas (GAs), and multiple Monitor Stations (MSs).

The GPS OCS utilizes GAs and MSs to monitor and control the GPS constellation and to receive GPS navigation data. Ground antennas contain equipment to receive a single GPS S-Band State-of-Health telemetry downlink and to uplink satellite commands and Navigation Uploads to a GPS satellite. Each MS contains a GPS L-Band Antenna and a twelve channel receiver which allows it to process the Navigation Telemetry downlink from multiple GPS satellites simultaneously.

Need for a Simulator

The Air Force has used high fidelity aircraft simulators for years to train pilots on new aircraft and mission operations. Until recently, the space segment of the Air Force had no similar capabilities. Several small satellite simulators have been developed to emulate different aspects of a particular mission but relatively few programs have the capability to simulate a complete, end-to-end mission with high fidelity. The high fidelity GPS constellation simulator gives the Air Force a powerful tool to train GPS satellite operators on command and control procedures.

Current Training Methods

For years, the GPS trainers have relied on "paper training" to educate new personnel on satellite operations. This paper training consists of actual satellite controller computer screens displaying "canned" telemetry for a scripted training scenario. The trainee must type in the proper commands for the script to match the "canned" telemetry. When the trainer wishes to train a particular anomaly, a hardcopy printout showing the anomalous telemetry screen is put in front of the trainee's screen and the trainee must then determine the anomalous telemetry and propose a corrective action. The trainer must then inform the trainee if he or she made the correct decision.

To augment this method, trainers have developed PowerPoint slides that look like the telemetry screens. When the trainee types a command, the trainer selects the appropriate PowerPoint slide showing the results of this command and displays it to the trainee.

These methods have several large drawbacks. First, the trainee gets no hands-on time on an actual system, a necessity for learning. Second, anomaly injection, recognition, and resolution are impossible to accurately train since the trainer is informing the crew that an anomaly exists by showing them the anomalous data on paper. Using this method, the user doesn't experience true interaction with the system. Finally, the trainer cannot anticipate all the user responses. If the student enters a command that was not anticipated by the trainer, the student will not see the accurate response to his or her command. Even with its drawbacks, this is the current method for the majority of the training.

Several years ago, the GPS controllers recognized the need for some sort of computer based training system. While they knew a high fidelity simulator was years away due to the time involved in a procurement cycle, they realized that a program aimed at obtaining minimum capability was a possibility. A stopgap measure was put into place to train with the existing system used for system test. This limited, low fidelity implementation gave the trainees hands on access to the hardware and software needed to command the satellites in a controlled environment.

GPS High Fidelity System Simulation

When the High Fidelity Satellite System (HFSS) was proposed, the goal was to produce a simulator so realistic that the crew could not differentiate between the simulator and the real constellation. To achieve this goal the entire GPS system, from end-to-end, needed to be simulated. The result is a simulator that models all satellite bus and payload subsystems as well as ground station hardware and environmental effects. This concept gives the trainers a tool to train the satellite controllers in the manner in which they will use the system. In addition to the system training concept, the high fidelity simulator allows the trainer to inject an anomaly to alter the state of the system without the student knowing. A great feature of this training environment is that the injection of anomalies requires the recognition of the failed subsystem by the trainee using the tools that would be available to him or her during live operations. These features promote the goal of "simulation so realistic that a GPS satellite controller cannot distinguish between simulation and live operations."

Simulation Engineer

The Simulation Engineer (SE) plays a crucial part in the training session. The SE initializes the simulator, executes the simulation, injects anomalies, and performs state saves.

Scripts and Initial Condition Sets

The SE must first decide what training objectives are to be accomplished. When this is decided, the SE must then build a training script using the simulator Human Computer Interface (HCI). The two components of a script are the directive and the task. These elements are defined here.

Directive	A command sent to the simulator to elicit some kind of change in the simulation. Directives can be chosen from the Builder Menu.
Task	A collection of directives, typically anomalies, all of which define a specific training objective.

Script	A collection of tasks, each task is designed to complete one training objective.

The directives are typically anomalies devised to cause the simulator to act in a non-nominal state. There are nearly 100 anomalies specified for all the different simulated spacecraft subsystems, ground equipment, and GPS environment. Several examples include Command Reject, Battery Heater Failure, and Thruster Misalignment. Directives are then built into a task, which typically defines a list of anomalies to be applied to a particular subsystem at a given time. For example, the SE could train attitude control procedures by building a task to fire a thruster and fail a reaction wheel. These two directives would be issued simultaneously resulting in a situation requiring attitude control response from the operator. Finally, a set of tasks are chosen and placed into a script, which is then executed to achieve the training objective. With a set of tasks and directives part of the baseline HFSS, a trainer can build a training script in a matter of minutes. The other main item that needs to be created by the SE is the Initial Conditions (IC) set. The following list defines an IC set.

- Start time of the simulation
- Orbital configuration of the constellation
- Number and types of spacecraft and ground stations
- Model parameters for each model in each domain chosen.

To understand the amount of data in a typical IC set, consider the following. The HFSS is required to simulate a 30-satellite constellation with all spacecraft being propagated at the same time. Within each simulated satellite, there are approximately eleven separate subsystems propagating, each with their own components at an even lower level. For each major satellite subsystem, there are approximately 100 items of 'State Data'; data used for modeling purposes. This amounts to roughly 33,000 state data items for the satellites alone. This state data is stored within the simulator database, and read in as the

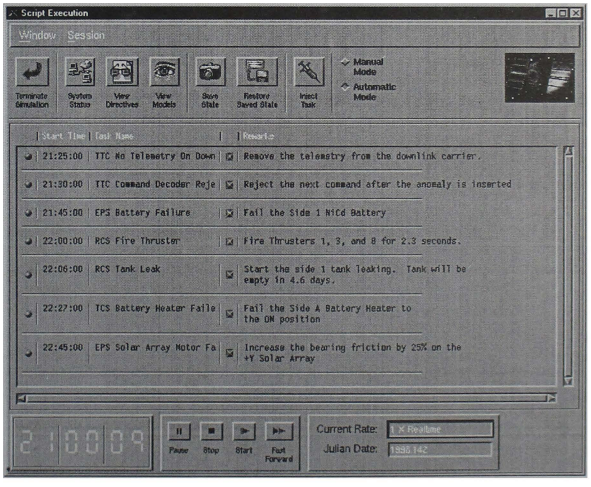


Figure 1 - Simulator Main Screen

IC set at initialization. The IC sets are changeable

by the SE and will generally be adjusted in between simulation runs if necessary.

Simulation Execution

Once a script and IC set have been chosen, the simulation models are initialized using the data contained in the chosen IC set. Next, the SE may execute the simulation simply by pressing the Start button shown in figure 1. Pressing this button causes the simulator to begin propagating. During a simulation run, the SE can observe the changing state data for a specific component on a specific satellite by pressing the View Models button shown in figure 1. After a series of menus where the SE chooses the satellite and the component he or she wishes to view, the screen shown in figure 2 is displayed. From here, the SE can view the state data for the selected model updating in real-time. The boxes to the right of the data indicate that this state data item can be altered through a process called Parameter Variation. By clicking on this box, the SE is prompted for a new value to type in. The model will then use this new value in its dynamic calculations. A great advantage of Parameter Variation is that other models that use the varied item will also be affected in their dynamic calculations thus producing second, third, and N^{th} order effects. This feature gives the SE a powerful tool during the training exercise. If the SE wishes for a particular component to act anomalously, and no anomaly has been previously defined for this component, then the SE can implement such behavior through parameter variation. A similar tool available to the SE is Telemetry Variation. This feature allows the SE to set any telemetry point to a particular value without changing the model dynamics. This feature is particularly useful when the SE wants the trainee to respond to the incorrect telemetry without influencing the other models in the system.

Training Scenarios

Once a script has been built and the simulation run has started, the training scenario begins.

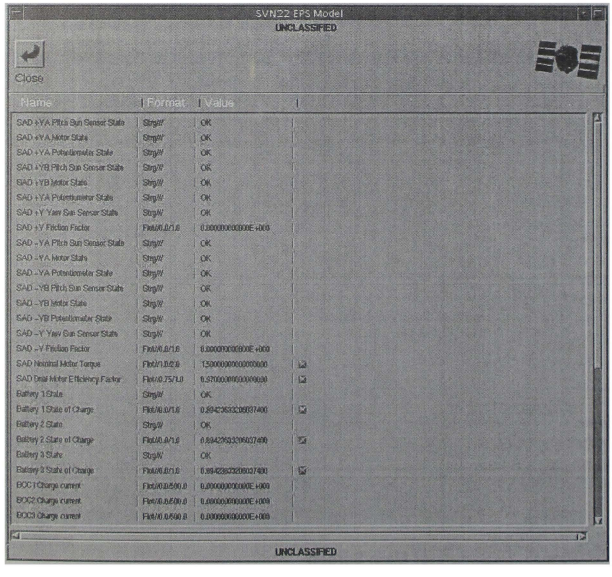


Figure 2 - Simulator State Data Screen

Once running, the SE can select the simulation to run in automatic mode or manual mode. In automatic mode, the tasks in the script are automatically executed at the time entered when the task was placed in the script. This mode is useful for canned training scenarios where the SE can readily anticipate the trainee's actions. Contrary to automatic mode, manual mode is much more dynamic and allows the SE more flexibility during the training session. Instead of the task executing at the predetermined time, the SE can click on the task when he or she wishes the task to execute. This allows the SE to be flexible when trainee responses are not anticipated or deviate from the prescribed plan.

Positional Training

A training option available with this simulator is to train individual crew positions. A crew position is best described as a role filled by one individual, performing a set of unique functions that contribute to the success of the mission. For the GPS mission, the positions include satellite bus specialists, payload specialists, ground equipment operators, and satellite pass controllers. For an example of position tasks, consider the following. The pass controller position initiates contact with the satellite and monitors the state of health of the satellite on the telemetry screens. With such a large amount of data flashing by in a relatively short amount of time, the pass controllers need to be trained to detect an anomaly quickly and respond with the appropriate corrective action.

Crew Training

While the HFSS provides an exceptional tool for positional training, the true power of this simulator lies in its crew training ability. The GPS mission requires that extremely accurate navigation data be sent to the ground. This task is accomplished by having a skilled crew that understands the system and can react quickly as a team. This skill level can only be achieved through comprehensive training and experience. For example, the navigation payload specialist must be trained to recognize when the payload on a specific satellite is not healthy. To do this, the ground equipment, the communications to the ground equipment, and the satellite's telemetry system must all be operating properly. Different personnel within the crew perform the task of ensuring that these functions are operating properly. Thus, for the GPS mission to be successful, a well trained crew is essential.

End-to-End Simulation

As eluded to earlier, many satellite simulators exist. Few, however, contain the complexity required for detailed crew training. HFSS fills this niche with an end-to-end simulation capability. Here, end-to-end means that using an actual ground station, commands can be generated, sent through simulated communications channels to simulated ground hardware where it is then sent to a simulated satellite. Here, the command is processed and the correct first through N^{th} order effects are simulated resulting in telemetry being sent to the ground and displayed to the controller. Again, many simulators exist that emulate the behavior of a certain subsystem or particular hardware component, but upon closer examination, many of these simulators do not communicate with each other and the effects produced by one are not included in the calculations of another.

Training Examples

The majority of the training conducted focuses on anomalous events in the GPS system. While non-anomalous training is performed, the demands on the crew are much less than when a problem arises. Therefore, to adequately train the crew so they are ready in a crises time, multiple anomalies are typically injected in a single script. While approximately 100 anomalies are available for training purposes, only a few are highlighted here. The following training scenario depicts several training objectives identified in the script in figure 1. By following and reacting to this script, the entire crew will be trained on anomaly detection and resolution of various subsystems.

When a student first starts the training session, he or she will be asked to start a contact between the ground segment and a particular satellite. Once the student has accomplished this task and has started to receive the downlink telemetry, the SE injects the task to remove the telemetry from the downlink carrier, followed by the task to reject the next command sent to the satellite. This causes the telemetry on the student's screens to stop updating. A quick look at the ground antenna equipment data informs the student that the antenna is still locked on a signal, there just is no

data modulated on that signal. The trainee must then reference the documentation that describes corrective actions. This document likely instructs the student to switch transmitters to remedy the problem. Upon sending the command to switch transmitters, the student notices that nothing happens. Since the SE has rejected this command, nothing should happen. Furthermore, the student is still not receiving telemetry so he or she has no knowledge that the command was rejected. Suspecting a command reject, he or she re-transmits the command before conducting further troubleshooting. With the second transmission, the student notices that telemetry again updates and the demodulation hardware again is locked on the satellite's telemetry signal.

Next, the SE issues the battery failure directive. There is little the student can do other than recognize the decrease in energy storage capability and keep the spacecraft load to a minimum when in eclipse.

Now, the SE sends the fire thruster directive to burn thrusters 1, 3, and 8 for 2.3 seconds. Since these thrusters are not opposing each other, a torque is imparted on the satellite. The momentum wheels controlling the spacecraft spin up to remove the applied torque. This action results in a greater spacecraft momentum, something that should be detected by the Spacecraft Bus specialist. Since the student has no knowledge of the actual orbit of the spacecraft, he or she must discern the effects on the spacecraft orbit through alternate channels. This channel is the navigation payload specialist. While monitoring the accuracy of the navigation signal, the payload specialist notices a change in the navigation accuracy. Using multiple diagnostic tools, the navigation payload specialist can detect the orbital error and prescribe the sequence of thruster firings to put the satellite back in the correct orbit.

Next, the SE injects the tank leak anomaly. This anomaly is quite easy to detect since the tank pressure is an item closely watched. Again, when this anomaly occurs, the controller can do little except isolate the tank from the rest of the reaction control system. This will prevent mass from being depleted from the other propellant tank.

Finally, the SE injects the battery heater failure and the increase in solar array bearing friction anomaly. Here, the student will need to recognize the battery temperature increasing beyond the normal limits. To correct this situation, the student must disable the failed battery heater. It is interesting to note that if this anomaly is undetected, the HFSS will propagate the anomaly to the other subsystems resulting in degraded system performance. For the increase in bearing friction, the student must recognize the increase in spacecraft momentum as the solar array drive motor imparts additional torque on the satellite to move the solar array. To correct this problem, the student simply sends the command to use the backup solar array drive motor.

As can be seen by the above scenario, many facets of GPS satellite operations can be trained with a relatively small number of anomalies. Here, the training script contains only seven anomalies and the entire crew was needed to detect, diagnose, and correct these problems. With a list of approximately 100 anomalies, it is easy to see how the HFSS can be utilized to train the GPS crews on nearly every possible GPS scenario.

Recommended Future HFSS Applications

During combined military training exercises, the GPS signal is relied on heavily for precise navigation and positioning. During these joint training scenarios, it would be valuable to know how a decrease in performance of GPS assets effects the other portions of the military. For example, if a particular GPS satellite is set unhealthy due to some payload anomaly, it would be beneficial to know how the accuracy of aircraft navigation, missile guidance, and ground troop positioning is affected. Where this information is typically only roughly computed for training and exercise purposes, the HFSS can compute what the exact impact to any user in any location will be, given a GPS anomaly that affects the navigation message. Not only can the magnitude of the effect be seen, but also the reaction of the pilots, missiles, and ground troops to the degraded signal. Additionally, if a GPS monitor station is removed from the system due to an anomaly, the impact on the GPS system's Kalman filter can be discerned. This impact ultimately affects any user, military or civilian, by generating a less accurate navigation signal. HFSS is the only simulation system that has the level of complexity to accurately compute these effects. By involving HFSS in these joint training exercises, the real

effect of degraded GPS performance can be ascertained.

Summary

Like their aircraft counterparts, satellite simulators strive to increase the level of fidelity to real world operations, increasing satellite system expertise, decreasing training time, and decreasing training costs. As stated earlier, the goal of the HFSS is to be so realistic that a controller cannot distinguish between the HFSS and the actual constellation. This goal was realized during formal system demonstration when a qualified crewman forgot for a moment that he was in contact with a simulated satellite and was worried that a particular command sequence could possibly cause harm to the satellite.

Acknowledgements

Ted Driver	Space and Environment Simulation Engineer for LM-MS
William Gregg	Database developer and administrator for LM-MS

AIAA-99-4120

DYNAMICS AND CONTROL OF A NOVEL RADIO TELESCOPE ANTENNA

Meyer Nahon*
Space and Subsea Robotics Laboratory
Department of Mechanical Engineering, University of Victoria
P.O. Box 3055, Victoria, B.C., Canada V8W 3P6

Abstract

This paper presents a mathematical and computer model of the triple-tethered aerostat proposed to support the receiver in a very large radio telescope antenna. The dynamics model considers the system as a set of three (or more) tethers, attached at fixed points on the ground, which come together at a spherical confluence point in which the receiver is located. Also attached to the confluence point is a leash tied to an aerostat overhead. The aerostat provides the necessary lifting force to keep the system aloft. At the ground attachment points, winches pull on the tethers to maintain the confluence point at its desired location. To investigate the performance of this system, we assembled a numerical model of its equations of motion. The tethers and leash were discretized into a number of elements using a lumped mass approach. The effects of cable stiffness, internal damping, gravity and aerodynamic drag as well as winds and turbulence were included in the model. The spherical confluence point and aerostat were modeled to include the effects of aerodynamic forces, as well as gravity and buoyancy. A controller was then developed to control the tether lengths in response to errors in the receiver position from a desired location. The complete system of almost 200 simultaneous first-order differential equations was formulated and solved numerically using a fourth-order Runge-Kutta integration scheme. Numerical experiments on this system indicate that the system can be accurately controlled in the presence of disturbances and that the concept warrants further study.

Introduction

Radio astronomers from around the world[1] have focussed recently on the need for a new radio telescope that would enable the direct observation of the formation and evolution of galaxies from gases in the universe.[2] To do this, requires an increase in sensitivity of roughly two orders of magnitude over existing radio telescope arrays---to a collecting area of 10^6 m^2 and is thus dubbed the Square Kilometer Array (SKA). One such conceptual design originates from the National Research Council of Canada's Herzberg Institute of Astrophysics and consists of an array of about 30 very large antennas, each of about 200-300 m diameter. The proposed novel antenna design,[3] a single one of which is depicted in Figure 1, is called the Large Adaptive Reflector (LAR).

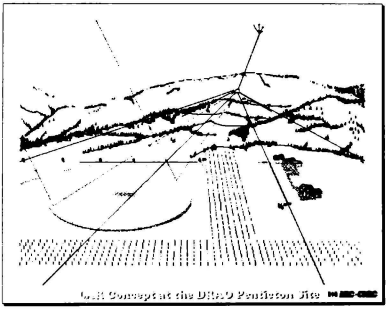


Figure 1. The Proposed Large Adaptive Reflector[3]

The LAR design is based on the premise that building a large scale steerable radio antenna will require new concepts in the design of structures to achieve the required strength and stiffness at a reasonable cost. The focus of each reflector will consist of a receiver held aloft at an altitude of R = 500 m by a tethered aerostat. To fulfil its function, the receiver must be positioned accurately at points on the surface of a hemisphere of radius R, centered at the center of the reflector. This will be accomplished by adjusting the tether lengths. It

* Associate Professor, Senior Member AIAA

Copyright © 1999 The American Institute of Aeronautics and Astronautics, Inc.
All rights reserved.

is also envisioned that the variable-length tethers will allow some measure of control of the receiver position in response to disturbances such as wind gusts.

A key issue which will determine whether the LAR design is feasible is that of positioning accuracy of the receiver. Among the questions which must be answered are: how much will typical winds and gusts disturb the receiver position ? and how effective might the tether winches be at overcoming these effects ?

Tethered and moored aerostat systems have received limited attention in the literature[4,5] and these have usually consisted of a large streamlined aerostat constrained by a single tether. These studies have focussed on the open-loop (uncontrolled) behavior of such systems. One key issue in modeling these systems is that of determination of the aerodynamic characteristics of the aerostat.[6,7] The aerostat design for LAR has not yet been finalized and may be spherical or streamlined. Some prior work at the National Research Council of Canada[8] dealt with a triple-tethered aerostat system, but focussed primarily on its static or steady-state performance.

Since a prototype would be prohibitively expensive to construct at this stage, a mathematical model and a computer simulation were assembled to study the dynamics and control of the system. These may be used for proof of concept studies and as design tools. A model of the planetary boundary layer mean wind and turbulence is implemented to investigate the effect of these disturbances on the system. A range of typical strong winds from a range of azimuthal directions are used and their effect is quantified with the receiver at a variety of zenith and azimuth angles (i.e. different receiver positions on its hemispherical trajectory). The effectiveness of the tether winches at controlling the receiver position is then evaluated using a simple controller consisting of independent PID loops on each tether.

Model Overview

The complete tethered aerostat system is shown diagrammatically in 2-D in Figure 2. The model consists of the following elements: (a) an aerostat of 40 KN gross lift, (b) a *leash* joining the aerostat to the receiver below it, (c) the receiver which is enclosed in a spherical shell, and (d) three elastic tethers from the ground to the common attachment point at the receiver. The aerostat is modeled as a single mass at the upper node of the leash, subject to buoyancy, aerodynamic drag (generated by winds and gusts), and gravity. Added mass of the aerostat is included, since it is large. The leash and tethers are modeled as a series of discrete cable elements connected by frictionless revolute joints. This effectively neglects the effects of bending stiffness since the dominant forces are due to tension. The mass of each element is lumped into its respective end nodes. The payload is modeled as a spherical point mass subject to gravity and aerodynamic drag.

The forces in the cable model are broken down into two types: internal and external forces. Forces generated within the cable are called internal forces and are due to axial stiffness and internal damping. Forces exerted on the rope by the environment are external forces, and consist of the aerodynamic drag and gravitational forces. The equations for internal forces and the drag forces are developed in an elemental body-fixed frame, while the equations for the gravitational forces are developed in an inertial frame. The motion equations are written in the inertial frame, and thus all forces must be transformed into that frame prior to inclusion in these equations.

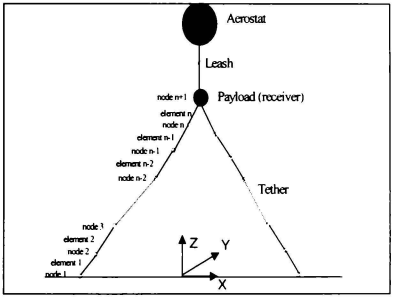


Figure 2. Discrete Implementation of Tethered Aerostat System

Cable Model

Two distinct types of orthogonal reference frame are used in the development of the mathematical model, an inertial reference frame and a series of body fixed reference frame placed at the nodes along the cables. The inertial reference frame (X, Y, Z) is defined with its origin located at the ground, at the center of the three tether attachment points. The X axis is in the horizontal plane and directed from the origin of the inertial frame to the base attachment point of Tether #1. The Z axis is vertical and positive upward, as shown in Fig. 2. Finally, the Y axis completes a right-handed coordinate system. The elemental body-fixed frame (p_1, p_2, q), is defined relative to each cable element, where p_1 and p_2 are the local normal and binormal directions, while q is the local tangent to the cable.

The solution of continuous cable models subject to general external forces is usually not possible and it is therefore considered more expedient to use a discrete model. The present work applies a lumped mass model with which we have a great deal of experience. This model has been validated for a variety of underwater systems with excellent agreement with in-field measurements.[9,10,11]

Kinematics

The position of each node is described with respect to an inertial reference frame, by a three-component vector $\mathbf{r}^i = [r^i_X \;\; r^i_Y \;\; r^i_Z]^T$. Each cable element is considered to be a straight elastic element, subject to forces at its end points. This method of modeling allows each cable element to possess distinct properties, such as density and stiffness.

The orientation of each cable element is represented using a Z-Y-X (ψ, θ, ϕ) Euler angle set. These three successive rotations align the inertial frame with the i-th body frame. Since the torsion of the cable is not included in the model, the ψ rotation about the inertial Z axis is constrained to zero. The resulting orthogonal rotation matrix which transforms vectors from the body-fixed frame of the i-th element to the inertial frame is denoted as[10]

$$\boldsymbol{R}^i_{IB} = \begin{bmatrix} \cos\theta^i & \sin\theta^i \sin\phi^i & \sin\theta^i \cos\phi^i \\ 0 & \cos\phi^i & -\sin\phi^i \\ -\sin\theta^i & \cos\theta^i \sin\phi^i & \cos\theta^i \cos\phi^i \end{bmatrix} \qquad (1)$$

The Euler angles can be calculated from the coordinates of the appropriate nodal end points. For example, consider the i-th cable element which is bounded by nodes i and i-1, shown in Fig. 3.

When expressed in the body frame, the only non-zero component of the vector $\mathbf{r}^i$-$\mathbf{r}^{i-1}$ is in the tangential direction. In component form, we have:

$$\boldsymbol{R}^i_{IB} \begin{Bmatrix} 0 \\ 0 \\ l^i \end{Bmatrix} = \begin{Bmatrix} r^i_X - r^{i-1}_X \\ r^i_Y - r^{i-1}_Y \\ r^i_Z - r^{i-1}_Z \end{Bmatrix} \qquad (2)$$

where l^i, the length of the i-th element, is given by

$$l^i = \sqrt{(r^i_X - r^{i-1}_X)^2 + (r^i_Y - r^{i-1}_Y)^2 + (r^i_Z - r^{i-1}_Z)^2} \qquad (3)$$

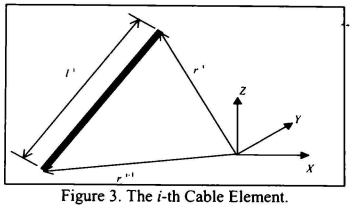


Figure 3. The i-th Cable Element.

Substitution of (1) into (2) results in the following set of equations[10]

$$\begin{aligned} l^i \sin\theta^i \cos\phi^i &= (r^i_X - r^{i-1}_X) \\ -l^i \sin\phi^i &= (r^i_Y - r^{i-1}_Y) \\ l^i \cos\theta^i \cos\phi^i &= (r^i_Z - r^{i-1}_Z) \end{aligned} \qquad (4)$$

which can be combined to yield

$$\theta^i = \text{atan2}\,(r^i_X - r^{i-1}_X, r^i_Z - r^{i-1}_Z) \qquad (5)$$

The solution for ϕ^i can then be found in either of two ways, depending on which is more numerically stable:

$$\phi^i = \text{atan2}\left[-(r^i_Y - r^{i-1}_Y), \frac{(r^i_Z - r^{i-1}_Z)}{\cos\theta^i}\right] \quad \text{if } \cos\theta \geq \sin\theta^i$$

$$\phi^i = \text{atan2}\left[-(r^i_Y - r^{i-1}_Y), \frac{(r^i_X - r^{i-1}_X)}{\sin\theta^i}\right] \quad \text{if } \cos\theta < \sin\theta^i \qquad (6)$$

Thus, (5) and (6) are our desired relations which allow us to find an element's Euler angles from its nodal coordinates.

Internal Forces

The internal forces acting within an element are due to its viscoelastic properties. These are represented schematically in Fig. 4.

The tension in the cable due to its structural stiffness is considered to act only in the tangential direction and is modeled by a linear function relating tension and strain

$$T^i_q = AE\varepsilon, \qquad \varepsilon = \frac{l^i - l^i_u}{l^i_u} \qquad (7)$$

where l^i_u is the unstretched length of the i-th element, A is its cross sectional area, E is the effective Young's modulus of the cable, and ε is the strain. The tension is set to zero if the strain becomes negative. However, this condition was never reached in the cases investigated since the aerostat buoyancy is very large in relation to the weight of the system aloft.

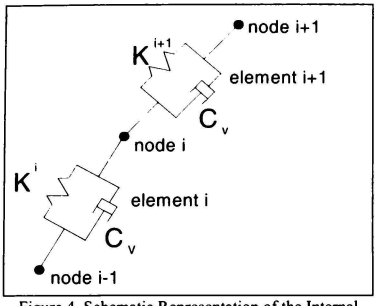


Figure 4. Schematic Representation of the Internal Forces

The friction between the braids of the cable tends to create a damping effect. This effect is assumed to be linear with the following relationship between strain rate and damping force

$$P_q^i = C_v(v_q^i - v_q^{i-1}) \tag{8}$$

where v^i_q is the velocity of the i-th node in the tangential direction.

External Forces

The external forces acting on the cable element are those due to aerodynamic drag and gravity. The drag forces acting on the cable element can be calculated according to[10]

$$D_{p1}^i = -(\frac{1}{2}\rho C_d d_c l_u^i) f_p \left|\mathbf{v}^i\right|^2 \frac{v_{p1}^i}{\sqrt{(v_{p1}^i)^2 + (v_{p2}^i)^2}}$$

$$D_{p2}^i = -(\frac{1}{2}\rho C_d d_c l_u^i) f_p \left|\mathbf{v}^i\right|^2 \frac{v_{p2}^i}{\sqrt{(v_{p1}^i)^2 + (v_{p2}^i)^2}}$$

$$D_q^i = -(\frac{1}{2}\rho C_d d_c l_u^i) f_q \left|\mathbf{v}^i\right|^2$$

where $\mathbf{D}^i$ is the aerodynamic drag force represented in the body-fixed frame with components D^i_{p1}, D^i_{p2} and D^i_q; ρ is the local density of air; d_c is the cable diameter; C_d is the normal drag coefficient of the cable; and $\mathbf{v}^i$ is the local velocity of the geometric center of the i-th cable element with respect to the surrounding air, with components v^i_{p1}, v^i_{p2}, and v^i_q. These velocities must account not only for the motion of the cable element, but also the motion of the surrounding air (to be discussed in a later section). In each of the above equations, the drag coefficient is modified by an appropriate loading function, f_p or f_q which are functions of η, the relative angle between the i-th element and the incident fluid flow. These loading functions account for the nonlinear breakup of drag between the normal and tangential directions. Driscoll and Nahon[9] surveyed several papers[12,13,14] which studied marine cable in a variety of towing conditions and arrived at the following synthesis of these models

$$f_p = 0.5 - 0.1\cos\eta + 0.1\sin\eta - 0.4\cos 2\eta - 0.011\sin 2\eta$$

$$f_q = 0.01(2.008 - 0.3858\eta + 1.9159\eta^2 - 4.16147\eta^3 + 3.5064\eta^4 - 1.187299\eta^5)$$

where η is expressed in radians and $0 \le \eta \le \pi/2$. The relative velocity of the flow over the geometric center of a particular cable element is found by averaging the relative velocities of its two end nodes, where the relative velocity of the air over node i is a function of the air velocity as well as the velocity of the node:

$$\left\|\mathbf{v}^i\right\| = \sqrt{\left(U_X^i - \frac{dr_X^i}{dt}\right)^2 + \left(U_Y^i - \frac{dr_Y^i}{dt}\right)^2 + \left(U_Z^i - \frac{dr_Z^i}{dt}\right)^2}$$

where U_X^i, U_Y^i, and U_Z^i are the local air velocity components at node i due to the mean wind and air turbulence. Once the drag for elements i and $i+1$ are calculated, half of each value is applied to the i-th node which joins the two elements.

Finally, the equation for the net gravitational force acting on a cable element in the inertial frame is:

$$W^i = \rho_c \frac{\pi d_c^2}{4} l_u^i g \tag{10}$$

where ρ_c is the density of material of the element; and g is the acceleration due to gravity.

Aerostat and Payload Models

In the present work, the aerostat is assumed to be spherical. Although a streamlined aerostat of equivalent lift would have significantly lower drag, and therefore be less affected by the local wind, it would also be much more expensive to purchase and operate. It was therefore preferred to first investigate the feasibility of using a spherical aerostat, while leaving the streamlined aerostat as a future backup option, should the spherical aerostat not be controllable to the required precision.

The payload located at the upper confluence point of the three tethers contains the antenna receiver and is assumed to be spherical in shape. Thus, for both the aerostat and the payload, we can write

$$D = \frac{1}{2}\rho C_d \frac{\pi d^2}{4}|\mathbf{v}|^2 \qquad (11)$$

where D is the aerodynamic drag force; d is the aerostat or payload diameter; C_d is the normal drag coefficient; and $\mathbf{v}$ is the local velocity of the geometric center of the aerostat or payload, relative to the surrounding air. The variation in drag coefficient of a spherical object from 0.4 to 0.15, depending on the Reynolds number of the flow,[15] was included. Once the drag force is calculated, it is decomposed into its inertial frame components for later incorporation into the motion equations.

The added mass of the aerostat and payload were included in the model since they are large. These were calculated as half the displaced air mass of the corresponding spheres.[16]

Assembly of the Motion Equations

Once all the internal and external forces described above have been expressed in the common inertial frame, the equations governing the motion of each node can be written as

$$\mathbf{M}^i\ddot{\mathbf{r}}^i = (\mathbf{T}^{i+1} + \mathbf{P}^{i+1}) - (\mathbf{T}^i + \mathbf{P}^i) + \frac{1}{2}(\mathbf{D}^i + \mathbf{D}^{i+1} + m_c^i\mathbf{g} + m_c^{i+1}\mathbf{g}) \qquad (12)$$

where $\mathbf{T}^i$ and $\mathbf{P}^i$ are the elastic and damping force vectors generated in the i-th element; $\mathbf{D}^i$ is the aerodynamic drag force vector on the i-th element; $m_c^i\mathbf{g}$ is the gravitational force acting on the i-th element. The drag, weight and buoyancy of the payload and aerostat are simply added to the appropriate cable nodes (at the lower and upper ends of the leash, respectively). All the node equations are then assembled and integrated simultaneously for $\mathbf{r}^i$ the position of node i in inertial space.

Mean Wind and Turbulence Model

A wind model was incorporated in order to determine its effect on the tethered aerostat system. It consists of a height-dependent mean wind profile on which are superimposed turbulent gusts which also vary with height. The mean wind U at height h was represented by a power law profile[17] representing the earth's boundary layer

$$U = U_g\left(\frac{h}{h_g}\right)^{\alpha} \qquad (13)$$

where the power law exponent α = 0.19 was used to represent conditions in rural areas. A gradient height h_g = 500 m was used, at which the mean wind reaches its full speed of U_g.

Turbulent gusts were superimposed on this mean wind. These were generated with the desired gust statistical properties,[17] including turbulence intensity, scale length and spectra. The turbulence intensities σ_u, σ_v, and σ_w in the three orthogonal directions were

$$\frac{\sigma_u}{U} = \begin{cases} 0.1 & h \geq 300\text{ m} \\ 0.1 + 4.328\times10^{-9}(300-h) & h < 300\text{ m} \end{cases}$$

$$\frac{\sigma_v}{\sigma_u} = \begin{cases} 1 & h \geq 300\text{ m} \\ 0.7 + h/1000 & h < 300\text{ m} \end{cases}$$

$$\frac{\sigma_w}{\sigma_u} = \begin{cases} 1 & h \geq 300\text{ m} \\ 0.25 + h/400 & h < 300\text{ m} \end{cases}$$

while the three corresponding scale lengths L_u, L_v, and L_w were

$$L_u = 280\left(\frac{h}{h_g}\right)^{0.35}, \qquad L_v = 140\left(\frac{h}{h_g}\right)^{0.48}$$

$$L_w = \begin{cases} 0.35h & h \leq 400\text{ m} \\ 140 & h > 400\text{ m} \end{cases} \qquad (14)$$

and finally, the three corresponding spectra Φ_u, Φ_v, and Φ_w are taken from a von Kàrmàn model[17] with

$$\Phi_u(\Omega) = \left(\frac{\sigma_u}{U}\right)^2 U^2 \frac{L_u}{\pi} \bullet \frac{1}{[1+(aL_u\Omega)^2]^{5/6}}$$

$$\Phi_v(\Omega) = \left(\frac{\sigma_v}{U}\right)^2 U^2 \frac{L_v}{2\pi} \bullet \frac{1+\frac{8}{3}(aL_v\Omega)^2}{[1+(aL_v\Omega)^2]^{11/6}} \qquad (15)$$

$$\Phi_w(\Omega) = \left(\frac{\sigma_w}{U}\right)^2 U^2 \frac{L_w}{2\pi} \bullet \frac{1+\frac{8}{3}(aL_w\Omega)^2}{[1+(aL_w\Omega)^2]^{11/6}}$$

where Ω is the wave number and a = 1.339. The following procedure was adopted to generate each of the three gust components

1. Discretize the wave number axis of the spectrum Φ_j into N = 50 intervals, each of width $\Delta\Omega_i$ = 1.2 $\Delta\Omega_{i-1}$ i=1,...,N, with Ω_{i1}=0.0001 and $\Delta\Omega_1$ = 0.0001
2. Randomly choose a wave number Ω_i within each interval.
3. Randomly choose a phase angle φ_i from 0 to 2π for each component.
4. Evaluate $\Phi_j(\Omega_i)/U^2$
5. Evaluate $A_i/U = (2\,\Phi_j\Delta\Omega_i/U^2)^{1/2}$

The above steps are all implemented off-line. The last step must be implemented during the dynamics simulation and is a function of position and time.

6. Evaluate the gust speed:

$$v_g = \sum_{i=1}^{N} \frac{A_i}{U} U \cos(\Omega_i (x - tU) + \varphi_i)$$

where x and t are the position along the mean wind direction and time, respectively. The above equation implies the use of a frozen-field approximation[17] to relate temporal and spatial gust correlations.

It is expected that the above model will incorporate the dominant features of the turbulence statistics, though it does neglect the spatial correlations in gusts in the directions perpendicular to the mean wind velocity.

Winch Controllers

In this work, straightforward PID controllers were implemented since the intention was primarily to perform an initial comparison of the uncontrolled and controlled systems. The controller on each winch is independent of the other two winch controllers and operates on the position of the payload, which we presume to be accurately measured. A planar representation of the geometry is shown in Figure 5.

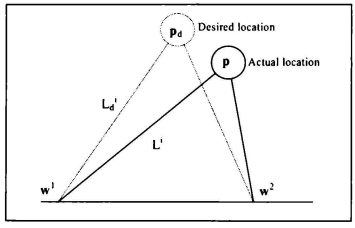


Figure 5. Control Terminology

The desired location of the payload is at $\mathbf{p}_d=[x_d\ y_d\ z_d]^T$ while its actual location is at $\mathbf{p}=[x\ y\ z]^T$. The location of the j-th winch (j = 1,..,3) is at $\mathbf{w}^j = [x_w{}^j\ y_w{}^j\ z_w{}^j]^T$. For each winch, we can therefore define the desired and actual distances to the payload as

$$L_d^j = \left\| \mathbf{p}_d - \mathbf{w}^j \right\|, \qquad L^j = \left\| \mathbf{p} - \mathbf{w}^j \right\| \tag{16}$$

Our winch controller can now operate according to

$$l_u^1 = l_{u0}^1 - k_d (\dot{L}_d^j - \dot{L}^j) - k_p (L_d^j - L^j) - k_i \int (L_d^j - L^j) dt$$

where l_u^j is the unstretched length of the first (lowermost) element in tether j, while l_{u0}^j is its initial length. The motivation behind this approach is that, if the distance from winch j to the payload is correct, then the payload lies on a sphere of radius L_d^j centered at the winch. If all three payload-winch distances are correct, then the payload lies at the intersection of those spheres which define the correct desired location of the payload in 3-D space.

Numerical Implementation

The second order differential equations given by (12) must now be solved by numerical integration. Each tether consists of n_t elements---and the last of these are connected at the leash---so that there are a total of $3(n_t+1)-2$ nodes in the tether system. The leash consists of an additional n_l elements, and, since the lowermost end node is common to the three tethers, a total of n_l nodes are added to the system, thus leading to a total of $3(n_t+1)-2+\ n_l$ nodes in the system. Each node is modeled by three second-order ODEs (one for each nodal degree of freedom). However, the lowermost node of each tether has no differential equations associated with it because its motion is prescribed. Furthermore, the aerostat and payload do not add any differential equations since their effect is lumped into the leash end nodes. In order to apply a numerical integration algorithm in standard form, each second-order ODE is rewritten as two first-order ODEs. This then leads to a total of of $18n_t-12+\ 6n_l$ first-order ODEs to represent the complete system.

The approach taken to solve the system of differential equations is taken from Reference 18. The particular integrator chosen is a fourth order Runge Kutta technique with fixed step size.[18] This method gives accurate solutions for a broad range of problems, though not necessarily with optimum efficiency. Because the system of differential equations is large (180 first order ODEs for n_t = *10, and* n_l = *2),* and the cable material is stiff (E=1.7×10^{10} Pa), a small time step had to be used to avoid numerical instabilities and ensure an accurate solution. Accuracy tests were made to ensure that an appropriate time step was used, ultimately leading to a time step of 1 ms.

Numerical Results

The particular configuration simulated was one in which the payload's desired position was at a fixed point on a hemisphere of radius R = 500 m whose center lay on the earth's surface. The winches were each located 1200 m from the hemisphere's center and configured symmetrically 120° apart. The desired location of the payload is specified by its *zenith* θ_{zd} and *azimuth* θ_{ad} angles, shown in Figure 6.

From this geometry, we find that the desired position coordinates of the payload are

$$\mathbf{p}_d = \left[R\cos\theta_{ad} \sin\theta_{zd} \quad R\sin\theta_{ad} \sin\theta_{zd} \quad R\cos\theta_{zd} \right]^T$$

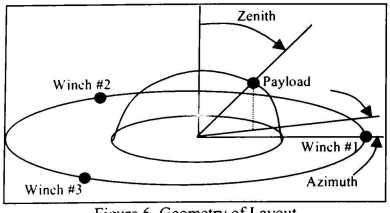


Figure 6. Geometry of Layout

The tether and leash consisted of Samson Spectron rope of 1.85 cm diameter, with a density of 840 kg/m^3 and a normal drag coefficient of 1.2. The 500 kg payload was contained in a sphere of 6 m diameter, while the spherical aerostat had diameter of 19.7 m, a mass of 610 kg, and a gross buoyancy of 40870 N.

The simulation described above was first validated by comparing its steady-state results against an independently-generated statics simulation.[8] All results matched within a fraction of a percent, both with and without steady winds (no turbulence).

A range of test cases at different azimuth and zenith angles was then run, for a variety of wind conditions. Two sets of results are shown here to highlight salient features of the system's performance. The first set of results compares the uncontrolled and controlled behavior of the system when $\theta_{za} = \theta_{az} = 0$ i.e., the desired payload position is directly above the center of the main reflector. The mean wind and turbulence are absent for the first 5 seconds, at which time U_g rises to 10 m/s, directed along zero azimuth. The payload motion and winch tensions are shown in Figures 7 and 8, respectively, with and without winch control. The motion is plotted as components in and out of the focal plane (the plane locally tangent to the 500 m hemisphere at the desired aerostat location) because only those errors in the focal plane are of primary importance. When $\theta_{za} = \theta_{az} = 0$, the focal plane is horizontal and the out-of-plane motion is purely vertical. The uncontrolled case makes apparent that the system responds little to high-frequency gusts and acts as a natural low-pass filter due to its huge scale. However, the error in the focal plane peaking at 1.5 m is unacceptable (though detailed ranges of acceptable motion have not yet been fully defined). When winch control is applied with k_d = 1 s, k_p = 5, k_i = 5 s^{-1}, the peak focal plane error is reduced to less than 5 cm.

The second test case shows the performance of the system for $\theta_{za} = 60°$, $\theta_{az} = 0$ primarily to highlight the degradation in system behavior at large zenith angles. The motion errors and winch tensions are shown in Figs. 9 and 10, respectively. It was found that at large zenith angles, the controller gains had to be reduced to avoid instabilities, indicating that the system is inherently less stable. The motion of the uncontrolled system at large zenith angles supports this interpretation. When compared to the preceding test case, the system oscillations appear much less damped. For the large zenith angles, it was also difficult to find a stable initial condition, and this is indicated by the non-zero motion of the system at t = 0. The controller, with k_d = 4 s, k_p = 1, and k_i = 1 s^{-1}, reduces the motion of the system, but not as significantly as in the first test case.

Other test cases, not shown here, revealed that the system is not very sensitive to the azimuth angle of the payload, and somewhat sensitive to the mean wind direction. Based on these findings, we expect that a long-term control solution will have to be adaptive in nature and change its characteristics at least with the payload zenith angle. It is also likely that a model-based controller could be applied to good advantage.

Conclusions

The computer simulation described here provides an efficient and cost effective method of determining the effects of different wind conditions and aerostat configurations on the tethered aerostat system described. The key findings of the study are: (a) the system is relatively insensitive to turbulent gusts and only responds to the lowest frequency gusts; (b) the system is significantly less well behaved (i.e. less stable) at large zenith angles, and (c) simple controllers developed in the course of this short study appear to be capable of reducing the motion response of the confluence point within a few centimeters of its desired location.

Future work will include incorporation of a streamlined aerostat model, incorporation of more than three tethers, improvement of the turbulence model, and an investigation of adaptive model-based controllers. Experimental validation at model scale would also be advantageous to confirm the computer model results.

Acknowledgements

This work was supported financially by the Natural Sciences and Engineering Research Council and the National Research Council.

References

[1] 'Memorandum of an Agreement to Cooperate in a Technology Study Program Leading to a Future Very Large Radio Telescope', signed by participants from Australia, Canada, China, India, the Netherlands, the U.S.A. http://www.nfra.nl/skai/memorandum.htm

[2] R. van de Weygaert and T. S. van Albada, 1996, 'New Challenges for Cosmology', *The Westerbork Observatory. Continuing Adventure in Radio Astronomy*, Eds. E. Raimond and R. Genee, Astrophysics and Space Science Library Volume 207, pp. 225--259.

[3] T. H. Legg, 1998, 'A Proposed New Design for a Large Radio Telescope', *Astronomy and Astrophysics Supplement*, v.130, p.369-379.

[4] S. P. Jones and J.A. Krausman, 1982, 'Nonlinear Dynamic Simulation of a Tethered Aerostat', *AIAA Journal of Aircraft*, Vol. 19, No. 8, pp. 679—686.

[5] S. P. Jones, 1987, 'Nonlinear Dynamic Simulation of a Moored Aerostat', AIAA Paper 87-2505.

[6] S. P. Jones and J. D. DeLaurier, 1983, 'Aerodynamic Estimation Techniques for Aerostats and Airships, *AIAA Journal of Aircraft*, Vol. 20, No. 2, pp. 120—126.

[7] S. Badesha and S. P. Jones, 1993,'Aerodynamics of the TCOM 71M Aerostat', AIAA Paper 93-4036-CP.

[8] B. Veidt and P. Dewdney, 1996, 'Steady-State Stability Analysis of a Triple Tethered Balloon Platform', DRAO Future Radio Facilities Report #1.

[9] F. R. Driscoll and M. Nahon, 1996, 'Mathematical Modeling and Simulation of a Moored Buoy System', *Proceedings of Oceans'96*, pp. 517—523.

[10] B. Buckham, M. Nahon and M. Seto, 'Three-Dimensional Dynamics Simulation of a Towed Underwater Vehicle, *Proceedings of OMAE'99, the 18th International Conference on Offshore Mechanics and Arctic Engineering*.

[11] F. R. Driscoll, 1999, 'Dynamics of a Vertically Tethered Marine Platform', PhD. Thesis, University of Victoria.

[12] R. Folb and J. J. Nelligan, 1983, 'Hydrodynamic Loading on Armored Towcables', David W. Taylor Navel Ship Research and Development Center, DTNSRDC-82/116.

[13] M. J. Casarella, 1970, 'Survey of Investigations on the Configuration and Motion of Cable Systems Under Hydrodynamic Loading', *Marine Technology Society Journal*, Vol. 4, No. 4, pp. 27-44.

[14] Y. Choo, 1971, 'Hydrodynamic Resistance of Towed Cables', *Journal of Hydronautics*, Vol. 5, No. 4, pp. 126-131.

[15] B. W. McCormick, 1995, *Aerodynamics, Aeronautics and Flight Mechanics*, John Wiley & Sons.

[16] J. N. Newman, 1989, *Marine Hydrodynamics*, MIT Press.

[17] B. Etkin, 1972, *Dynamics of Atmospheric Flight*, John Wiley & Sons.

[18] W. H. Press, et al., 1992, *Numerical Recipes in C*, Cambridge University Press.

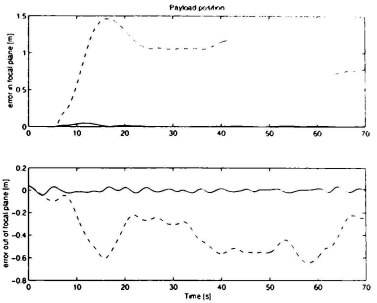


Figure 7. Payload Motion for $\theta_{z0} = \theta_{az} = 0$ With Control (solid line) and Without Control (dashed line)

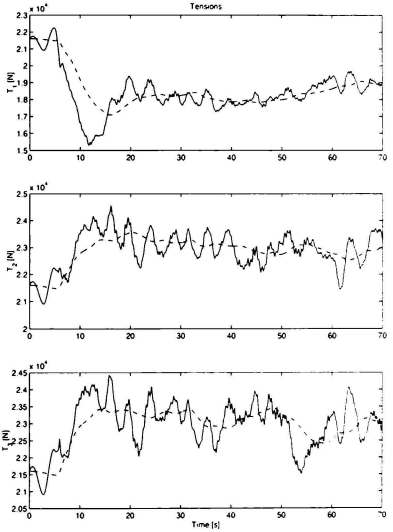


Figure 8. Winch Tensions for $\theta_{z0} = \theta_{az} = 0$ With Control (solid line) and Without Control (dashed line)

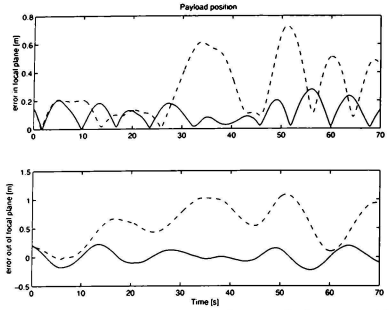


Figure 9. Payload Motion for $\theta_{z0} = 60°$, $\theta_{az} = 0$ With Control (solid line) and Without Control (dashed line)

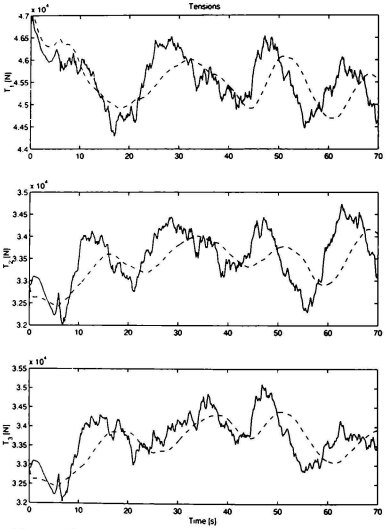


Figure 10. Winch Tensions for $\theta_{z0} = 60°$, $\theta_{az} = 0$ With Control (solid line) and Without Control (dashed line)

AIAA 99-4121

FLEXIBLE, INTUITIVE INTERFACE IN COMPUTATIONAL FRAMEWORK FOR DESIGN OF COMPLEX SYSTEMS

Nelson Thompson, Dynacs Engineering, Houston, Texas
Michael Tigges, National Aeronautics and Space Administration
Johnson Space Center, Houston, Texas

Abstract-- This paper presents a design methodology to implement an integrated computational framework for performing space mission design and analysis. This methodology arises from a need to modernize legacy analytical tools, preserving their usefulness into the next millennium. The underlying design principles for this method are modernization, customizability, intelligent data format parsing, and ease of utilization of an existing legacy tool set. The toolset currently being integrated using this method is in the Flight Mechanics Laboratory (FM Lab) at the Johnson Space Center (JSC). This method is the Advanced Tool Integration Environment (ATIE), and provides a concept for a fundamental element in the development of multi-site collaborative engineering environments.

Table of Contents

1. Introduction
2. The Problem
3. ATIE Description
4. Standard Data Formats
5. Integrating a New Tool
6. Configuring the Tools
7. The Control Interface
8. Status
9. Conclusion

1. Introduction

Existing methods to construct usable computational frameworks for collaborative engineering have focused upon two strategies: (1) implementing the mechanics of piping data between dissimilar analytical applications (typically over networks)[1], and (2) creating conceptual abstractions of the collaborative process itself, in an attempt to generate "intelligent" collaboration assistants[2,3,4]. Both of these strategies have demonstrated significant success. However, two common shortcomings reported are that the frameworks are limited to the specific engineering problem being solved (the context of the computational framework), and the effort required to set up or modify the computational framework is excessive.

Copyright © 1999 by the American Institute of Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner.

Whereas strategy (1) could be termed a pure "bottom up" approach, and strategy (2) could be termed a pure "top down" approach, ATIE can be thought of as a "from the middle out" strategy. ATIE focuses on developing a simple data and control interface that integrates the tools at one site, and makes that same interface accessible to other sites and to multi-site collaborative efforts.

With minimum effort it then becomes possible to "modernize" a generic legacy tool interface while maintaining the integrity of the original code. Additionally, working within the ATIE framework, a user may extend the toolset capability and automatically track evolving executables and corresponding user inputs. ATIE then becomes an integral aspect of a static and/or dynamic project build phase, which may incorporate data, tools and participants from multiple remote sites.

2. The Problem

The software tools for large collaborative engineering projects (like designing an aerospace mission to Mars) are typically composed of legacy applications developed independently over many years, without GUIs or regard for future integration. They have been validated at great expense, so modifying them is often not an affordable option. See Figure 1.

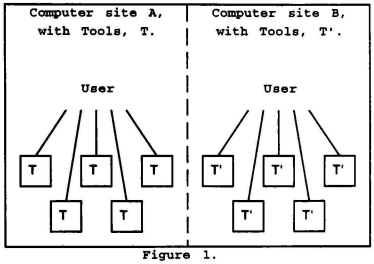


Figure 1.

Each of these tools solves one specific family (or a few related families) of problems. For example, one tool may calculate launch windows, another may determine optimum trajectories, others aid in the

design of the spacecraft itself and its subsystems, and yet others analyze the requirements for various modes of re-entry and landing. Solving an end-to-end mission design problem in a truly integrated fashion has been impractical, until now.

The approach to building a computation framework (CF) to effect the integration of these disparate tools, has been to attempt to integrate several specific tools at distributed sites to do one well-defined job. Such multi-site configurations, connected via the Internet, have been called Integrated Design Environments, or IDEs. See Figure 2.

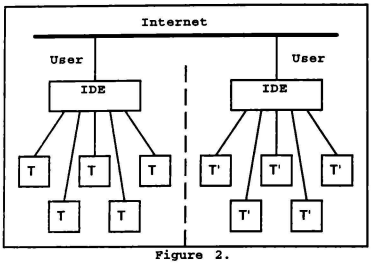


Figure 2.

Providing an *ad hoc* distributed IDE provides a demonstration of feasibility, but once accomplished, the context of the solution (the collaborative task being performed) is not easily modified to handle a different task. The configuration must be developed anew. Even given a sophisticated GUI as a development aid, this can be immensely non-trivial.

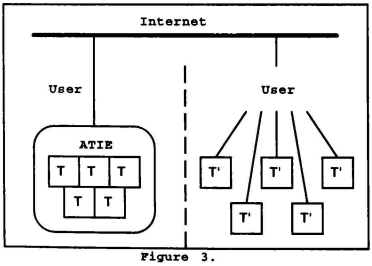


Figure 3.

The ATIE approach is to integrate many legacy tools at one site, so that (1) they can be flexibly configured, at reduced costs in terms of time and effort; and (2) a standard GUI, data format and control paradigm is made available to remote users over the Internet. The several tools incorporated into ATIE can be treated as one integrated toolset. See Figure 3.

This also provides a uniform interface for subsequent integration of those local tools with other tools at distributed sites. We believe this would effectively create a simpler means of integrating the FM Lab toolset with remotely distributed tools and data as part of a multi-site collaborative effort. See Figure 4.

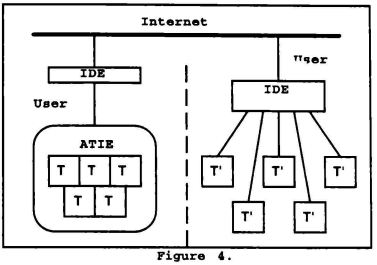


Figure 4.

Ideally, IDEs would be maximally flexible if all local sets of legacy analytical tools were first wrapped within an implementation of ATIE. See Figure 5.

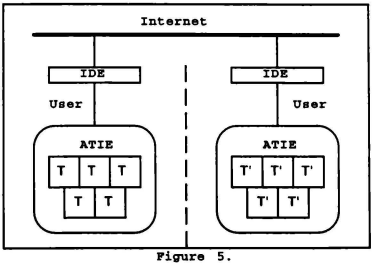


Figure 5.

This would provide a standardized, customizable, integrated environment for remote execution of local legacy toolsets. The diversity of the toolsets at the different centers (or nodes) remains intact. Only the interface for each ATIE node is standardized.

3. ATIE Description

The ATIE relies upon both common gateway interface (CGI) and an active graphic user interface (GUI)[5]. The use of CGI enables the selection of any common web browser as the foundation application with which to access the ATIE. The ATIE appears in the browser as a website with a number of "pages."

The active GUI consists of a set of custom-designed Java applets imbedded within the pages, that support data display, editing, and all control functions over the analytical tools themselves, including execution, subroutine editing, linking, iteration and termination. Navigating among the various interface controls ("pages") can be intuitively accomplished with the browser's functionality, thus handling a major portion of the user's interface to the ATIE, whether the user be local or remote.

The server-side core of ATIE consists of two principle functionalities: the Automated File Format Translator, AFFT; and the Local Applications Sequential Executor, LASE.

4. Standard Data Formats

A key element required for flexibility in a computational framework built around legacy tools is the ability to recognize and parse data inputs in a wide variety of formats, and to automatically translate data across different formatting schemes. Fortunately, data formats are constrained in general. This allows construction of an intelligent parser that reads an input file using a small number of formatting rules. This permits the display and editing of data in one common format while avoiding the need to re-certify the analytical tools or their data sets.

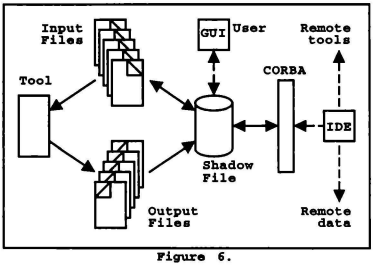


Figure 6.

AFFT provides an extendible, data-driven parsing engine capable of interpreting any common (or even most plausible) formats of ASCII text input. This data, along with its physical units and comments is stored in a standard, common format, the "Shadow" file. It is the Shadow file that the user edits via the GUI; that AFFT retro-parses back into the original format so that it can be input into the analytical application (tool); and that is transmitted to, and updated from, remote applications lashed together with a multi-site IDE. See Figure 6.

Translating all data into one standard format provides a single, uniform interface to the local user and the remote user. If the appropriate elements of the Shadow file are wrapped with a CORBA compliant interface, then a single, uniform data and control interface is presented to multi-site IDEs.

5. Integrating a New Tool

Given that AFFT is data driven, incorporating a new tool into the ATIE configuration becomes relatively easy. One of the ATIE GUI elements is a "wizard" that asks the user a series of questions about the data formats for the new tool. Here is a partial list of questions concerning the input format, plus a list of discrete answers from which the user may select:

- How are comments initiated?

  `/` `//` `--` `/*` `<`

- How are comments terminated?

  `newline` `*/` `>`

- How are items in a list delimited?

  `,` `;` `space`

- What is the equate symbol?

  `=` `:` `:=` `not allowed`

- How are parameter indices bounded?

  `()` `[]` `{}` `not allowed`

- How are physical units bounded?

  `{}` `:=` `not allowed`

- How are lines terminated?

  `;` `:` `#` `newline`

If the correct answer to a question isn't tacitly available, the user may manually enter it. For example, if comments are initiated with the octomorph ('#'), the user can enter that character. (This subsequently updates the Wizard, so that when the next tool is integrated, the octomorph will be automatically included in the list for that question.)

From the answers provided, AFFT builds a series of regular expressions (in the Unix sense), which are used in the AFFT Perl scripts to detect particular characters or formats within that tool's input and output files. In this manner, all information within the files can be parsed.

This also enables ATIE to retro-parse the edited data in the Shadow file back into the original text format

that the associated tool requires for input. Eventually, it may be advantageous to store all analytical data in a permanent Shadow database, generating specifically formatted text files only as necessary to feed the tools.

To complete the tool integration process, the user provides information on the tool's location and execution command, and an example input file. If the display of that data, and execution of the tool proceed without error, then the new tool is integrated into ATIE. From that point, the new tool can be used in concert with all the other ATIE tools.

When a new tool (or new tool version) is integrated into ATIE, the data files appropriate to that tool are associated by ATIE in its database with that tool. This becomes quite important for those tools that exist in several *ad hoc* versions or configurations.

For those analytical tools that must be relinked and recompiled to produce a custom executable appropriate to a specific analytical problem, this provides users with the opportunity to modify source code and build executables in a graphical environment. This auto-tracking feature is vital for a dynamic analytical environment with evolving toolsets.

6. Configuring the Tools

LASE, or Local Application Sequential Executor, is the second principle part of ATIE; it provides the ability to automate the sequential execution of tools that have been integrated into ATIE.

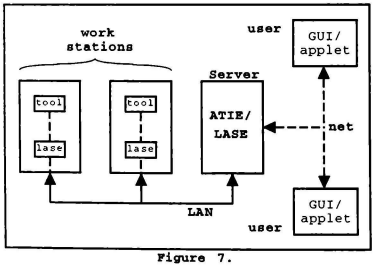


Figure 7.

LASE has three components: a Java applet that enables the user to define a particular sequence of tools, their input and output files, the specific parameters that are passed between tools, and enables the user to control the execution of the sequence (including any iteration loops) once it is set up; a Perl script on the server that generates the actual sequence execution scripts and runs them; and several Perl/Expect scripts that actually generate commands to the tools and pipe their outputs back to the server. See Figure 7.

Generating an intuitive interface for controlling the sequence of tool execution proved problematic. Defining such a sequence, with all the appropriate files, and the specific parameters to be passed between tools, and their valid ranges, can encompass an enormous amount of information. Entering this information in a list or form or script could prove daunting, if not overwhelming to the majority of users.

It was decided to provide a graphical representation of the execution process, based upon an old and revered design technique: the N-squared diagram,[2] which is represented in the LASE Java applet displayed in the user's browser. See Figure 8.

Each element of the applet (except for the lines) is an interactive component. Clicking on these components calls up segments of the control interface in overlying browser windows. Furthermore, when the sequence of tools is executed, the components change color to reflect the state of the execution.

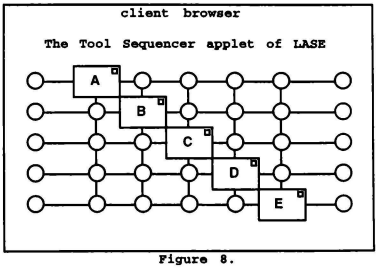


Figure 8.

The rectangles within the applet represent tools to be executed, and are ordered from top left to bottom right. Clicking on a rectangle enables the user to select the specific tool to be associated with that icon.

The circles, or "nodes," in the matrix immediately above the rectangles (tools) represent the "forward" interfaces between the tools. Clicking on a node with tool A to the left (same row) and with tool C below (same column) opens up a window which enables the user to select those specific outputs of tool A which should be fed into tool C.

Not all outputs of one tool are potential candidates for input into another tool. Rather than decide *a priori* which parameters from, say, tool A may be passed to tool C, we permit that interfacial information to be accumulated over time as the ATIE is used.

The user is initially presented with a list of interfacial parameters that have been previously utilized for a given (tool A/C) interface. If the desired parameters are not in that list, the user may drill down to the complete list of tool A outputs, and the complete list of tool C inputs, and freely select those which are to be associated. Any necessary format and/or unit conversions are chosen at this time. This process adds these new parameters to the initial tool A/C interface list.

If iteration is desired, the user clicks on the nodes in the matrix immediately below the rectangles, and defines the "feedback" interfaces between, say, tool C and tool B.

Clicking on the small squares within the rectangles enables the user to define the limits of iteration for that tool as either a specified number of loops, or as a conditional function of some value(s) of the tool's output parameters.

The left-most column of circles represent the default input files to the tools. They contain the parameter values not otherwise specified by the tool-to-tool interface lists.

The right-most column of circles represent the output files generated by the tools. After execution is complete, clicking on an output file icon will immediately open that file with the appropriate application. In some cases, the final outputs being produced may be plots, graphs, animations or computer "videos."

7. The Control Interface

Configuring a sequence of tools and their inputs for execution can be saved, and will be represented within the ATIE by data within a configuration database, that will instruct the LASE module in its role as "conductor" of the execution script.

For multi-site collaborations through an IDE, the appropriate elements of the configuration database will be wrapped with a CORBA compliant interface, so that execution status and commands are available; the IDE can then receive piped data, remote tool commands and execution status; and return data and commands in a common format.

8. Current Status

The current version of ATIE is operational and allows editing the input files for two of the more commonly used analytical tools within the FM Lab, and the execution of those two tools. The ATIE is being used by a few selected engineers in the Lab, and as a training aid for summer co-ops, with the intent of evaluating and evolving the basic interface as rapidly as possible. Ease of use and flexibility are the driving attributes of this evolution.

AFFT is complete and demonstrates reliability and robustness. LASE will be complete and operational in a few months; the applet front end is already complete and undergoing interface evaluation.

9. Conclusion

ATIE was initially proposed as a solution to an "in-house" engineering problem; namely, how to deal with an aging set of legacy tools that cannot be replaced, nor affordably updated, and for which the utilization knowledge base is dwindling at an accelerating pace.

In the course of searching for a solution, several potential solutions were observed: the IDEs which are in conceptual, early design or proof-of-concept stages of development. Though their interfaces were admirable, they did not expressly handle the legacy issue, and with their conceptual superstructures (project "abstractions"), were considered to be over-kill solutions for the FM Lab's problem.

However, it was clear that the Age of the IDE was upon us, and we looked anew at the ATIE as a possible element in the development of multi-site collaborative engineering environments. With this in mind, we conclude that:

(1) the ATIE provides a mechanism for translating data in any format to a standard format and back again, creating a "lingua franca" for passing data between dissimilar tools and data sets, thereby "pre-integrating" the tools at any one site;

(2) the ATIE separates the data parsing function from the tool configuration and execution functions, so that the latter can be done independently of the former, thereby presenting a uniform control paradigm for all tools integrated into ATIE at that site;

(3) the ATIE relies upon an additional, independent level of integration (CORBA-compliance) to expand the context to remote tools and data, presenting a single unified interface for inclusion into any

collaborative, distributive computational framework for the design and analysis of aerospace missions.

In this manner, the individual user who is only concerned with using and/or extending the tools at a single site (be that site local or remote), is presented with a single browser-based tool with an intuitive and flexible interface.

As larger, collaborative engineering environments (IDEs) become the norm, the ATIE is designed to interface with them.

REFERENCES

[1] Rogers, J.L.; Salas, A.O.; Weston, R.P.: "A WEB-Based Monitoring System for Multi-disciplinary Design Projects," AIAA-98- AIAA 23rd Aerospace Sciences Meeting, Reno, Nevada, January 1985.

[2] Lamarra, Norman; Dunphy, Julia: "Interactive Sharable Environment for Collaborative Spacecraft Design"

[3] Lamarra, Norman; Dunphy, Julia: "Web-Based Operation of Interactive Sharable Environment for Collaborative Spacecraft Design"

[4] Kalay, Yehuda E.: "P3: Computational Environment to Support Design Collaboration," published in Automation in Construction 8(1):37-48, November 1998.

REAL-TIME SIMULATION OF A FREE-FLYING ROBOTIC VEHICLE

Craig R. Carignan*
J. Corde Lane†
David L. Akin‡
Space Systems Laboratory, University of Maryland, College Park, MD 20742

The reaction forces caused by arm motion on a free-flying robotic vehicle can cause significant dynamic coupling when the vehicle is free-floating in space or underwater. This paper presents an inverse dynamics procedure based on the Recursive Newton-Euler Method for simulating this effect during human-in-the-loop operation. The methodology is applied to the Ranger Neutral Buoyancy Vehicle currently under development in the Space Systems Laboratory for microgravity simulation of robot servicing. Methods of countering this motion using reaction wheels and thrusters are examined for a class of free-flying robots.

Nomenclature

m	mass, kg
θ_i	angle of joint i, rad
p	3x1 position vector, m
v	3x1 velocity vector, m/s
$\dot{v}$	3x1 acceleration vector, m/s^2
${}^a\tilde{q}_b$	4x1 quaternion relating frame b to frame a
${}^aR_b$	3x3 direction cosine matrix relating frame b to frame a
ω_b	3x1 angular rate vector of frame b, rad/s
F	3x1 total force vector, N
N	3x1 total torque vector, $N-m$
f	3x1 link interaction force vector, N
n	3x1 link interaction torque vector, $N-m$
h	3x1 angular momentum vector, $N-m-s$
θ_a	Lx1 vector of arm joint angles, rad
I	3x3 inertia matrix, $kg-m^2$
$\hat{z}$	3x1 unit vector in z-direction
τ_a	Lx1 arm joint torque vector, $N-m$
${}^i(\cdot)$	designates quantity in frame i coordinates

Subscripts

v	vehicle
a	manipulator arm
w	reaction wheel

Coordinate Frames

0	manipulator base
b	vehicle body
C	center of mass
r	inertial reference

Introduction

Future on-orbit and undersea remote operations will require greater mobility and dexterity than currently possible with the Space Shuttle RMS and most submersible remotely piloted vehicles (RPV). This has led to the construction of free-flying teleobots with a suite of arms for performing various tasks. Operations range from maintenance tasks such as those performed during the Hubble servicing missions to deep sea recovery and salvage operations.

These robotic "vehicles" represent a merger of the latest in robotic and satellite/submersible technology, possessing free flight as well as robotic capability. Unlike industrial robots where the manipulator is attached to a rigid platform, arm motion can cause significant motion of the vehicle when it is free-floating in space or underwater. This paper develops the coupled arm/vehicle dynamics necessary to perform simulations of on-orbit or undersea operations when the vehicle is not docked to a target.

While it is expected that on-orbit servicing robots will possess a certain degree of autonomy, it is likely that many operations will still be performed teleoperatively. This human-in-the-loop interaction implies that the simulation software must operate in real-time, posing a significant challenge to software

*Research Engineer, Space Systems Laboratory, Dept. of Aerospace Engineering, University of Maryland, AIAA Member
†Graduate Research Assistant, Space Systems Laboratory
‡Associate Professor, Dept. of Aerospace Engineering, Director, Space Systems Laboratory, University of Maryland, AIAA Senior Member

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

developers. While a number of software packages are available to simulate complex dynamical systems, real-time simulations for the most part remain customized for the particular application.

In addition to the dynamics software, the control station plays a key role in the training simulator. A realistic graphical environment was generated on a Silicon Graphics O2 Workstation using MultigenII^{TM}, a real-time 3D animation package. The operator was able to input commands to the simulation using Graphical User Interfaces (GUI) as well as a variety of input devices including a pair of 3-axis hand controllers and a head tracking display.

The paper begins with an overview of free-flying robotic systems using an underwater vehicle developed in the Space Systems Lab as an example. The decoupled dynamics of the arm and the vehicle are then developed separately using current techniques. The modifications to the Newton-Euler Method necessary to accommodate a free-flying base are then developed. Next, a feedback regulator designed for satellite attitude control is considered for countering the effects of the arm torques on the base platform. The control station simulation environment and user input interfaces are then described. Finally, some examples of arm motion with and without the attitude control system engaged are used to illustrate the benefits of attitude stabilization.

Free-Flying Robotics Overview

The Space Systems Laboratory at the University of Maryland is in the unique position of studying operations both underwater and in space since it uses neutral buoyancy to simulate the microgravity environment. This work examines the operation of the Ranger Neutral Buoyancy Vehicle (NBV) which is a prototype of a robotic vehicle being considered for on-orbit satellite servicing within the next decade (see Figure 1). A non-free flying version of this vehicle, Ranger TSX (Telerobotic Shuttle Experiment), is slated to fly on the Space Shuttle sometime in the year 2001.[1]

The main components of the Ranger NBV are shown in Figure 2. The vehicle consists of three major sections:[2] the propulsion module (PM), the electronics module (EM), and the manipulator module (MM). The propulsion module serves as a platform for the eight vehicle thrusters as well as the battery boxes for powering the vehicle thrusters, electronics, and arms. The PM also houses five rotational buoyancy compensators and a vertical buoyancy compensator for fine adjustment of the vehicle buoyancy. The electronics module houses the main computer for the vehicle as well as instrumentation for determining vehicle attitude and rate. The manipulator module serves as the mounting platform for the vehicle's four manipulators: a seven-joint grapple arm which docks to the worksite and maneuvers the vehicle, a six-joint video arm to position a stereo camera pair for viewing operations, and two seven-joint dexterous arms for performing servicing tasks.

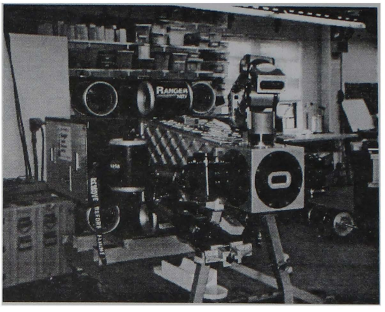


Fig. 1 Ranger NBV on deck.

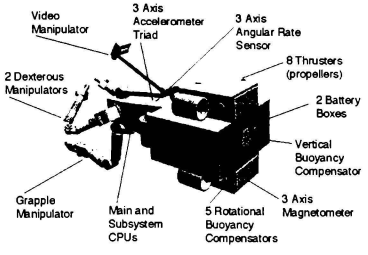


Fig. 2 Overview of Ranger NBV.

Perhaps the most difficult phase of an on-orbit repair mission is the "free-fly-to-grapple" stage in which the vehicle must fly to within close proximity of an ailing satellite and grab onto a docking fixture (see Figure 3). This has proven to be exceedingly difficult without the assistance of the thrusters (propellers) to stabilize the vehicle as the arm maneuvers into position. In practice, it was found that holding the attitude of the vehicle fixed while attempting to grapple the satellite was much more important than trying to hold a position.

Attitude stabilization on Ranger NBV was obtained by using thruster pairs to create moments to counter the reaction torques from the arms. While the use of thrusters for attitude stabilization would

be prohibitively expensive in space applications, it is quite plausible that control moment gyros or reaction wheels could be substituted. Since tasks would tend to require unbaised rotations, the reaction torques would not typically lead to the saturation of reaction wheels common in many satellite operations.



Fig. 3 Ranger NBV in free flight attempting to grapple handrail on satellite.

Decoupled Dynamics

To facilitate the derivation of the coupled vehicle/arm dynamics, the dynamics of the vehicle and arm will first be derived separately. The vehicle attitude dynamics development parallels that used for satellite dynamics. The manipulator dynamics are derived using the Iterative Newton-Euler Method which is commonly used in robotics.

Vehicle Dynamics

In this section, the general case of a rigid spacecraft rotating under the influence of body-fixed torquing devices is considered. For a rigid body with force F acting through the center of mass, the acceleration of the center of mass, $\dot{v}_C$, is given by

$$F = m\dot{v}_C \tag{1}$$

where m is the mass of the body, and F and $\dot{v}_C$ are both defined with respect to some inertial reference frame r. Equation (1) is referred to as "Newton's Equation".

The linear acceleration of the center of mass is related to the acceleration of the origin of the body reference frame b through the relation

$$\dot{v}_C = \dot{\omega} \times p_C + \omega \times (\omega \times p_C) + \dot{v} \tag{2}$$

where p_C is the position of the origin of the center of mass in the body frame b, v is the velocity of the origin of the body frame relative to frame r, and ω is the angular velocity of frame b with respect to r.

The angular acceleration $\dot{\omega}$ of a rigid body acted on by moment N is given by

$$N = {}^{C}I\dot{\omega} + \omega \times {}^{C}I\omega \tag{3}$$

where C is a frame with the origin at the center of mass, and ${}^{C}I$ is the inertia of the vehicle in frame C. Equation (3) is referred to as "Euler's Equation".

The addition of a reaction wheel to the satellite modifies (3) as follows[3]

$$N = {}^{C}I\dot{\omega} + \dot{h}_w + \omega \times ({}^{C}I\omega + h_w) \tag{4}$$

where the reaction wheel angular momentum, h_w, is given by

$$h_w = {}^{C}I_w\omega_w \tag{5}$$

and ${}^{C}I_w$ is the inertia of the reaction wheel in frame C.

Equations (4) and (1) represent the equations of motion for the vehicle without the arms attached. Note that the elements of these equations are typically defined in the body coordinates of the spacecraft since the inertias are constant in the body frame, and the torquing devices and thrusters are mounted to the vehicle.

Arm Dynamics

The Newton-Euler Method, in its recursive form, can also be used for deriving the dynamics of the manipulator. This algorithm employs the same methodology used to develop the vehicle dynamics, progressing from one link to the next. The recursive formulation consists of two parts: an outward recursion from the base to the end-effector to determine the link velocities, accelerations, and total forces/torques, and an inward recursion from the end-effector to the base to determine the joint interaction forces and torques. The equations for an L-joint revolute manipulator are as follows:[4]

Outward Iterations: $i : 0 \rightarrow L$

$$ {}^{i+1}\omega_{i+1} = {}^{i+1}R_i\,{}^{i}\omega_i + \dot{\theta}_{i+1}\hat{z} \tag{6}$$

$$ {}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}R_i\,{}^{i}\dot{\omega}_i + {}^{i+1}R_i\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\hat{z} + \ddot{\theta}_{i+1}\hat{z} \tag{7}$$

$$ {}^{i+1}\dot{v}_{i+1} = {}^{i+1}R_i({}^{i}\dot{\omega}_i \times {}^{i}p_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}p_{i+1}) + {}^{i}\dot{v}_i) \tag{8}$$

$$
\begin{aligned}
{}^{i+1}\dot{v}_{C_{i+1}} &= {}^{i+1}\dot{v}_{i+1} + {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}p_{C_{i+1}} + \\
&\quad {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}p_{C_{i+1}}) \quad (9)\\
{}^{i+1}F_{i+1} &= m_{i+1}{}^{i+1}\dot{v}_{C_{i+1}} \quad (10)\\
{}^{i+1}N_{i+1} &= {}^{C_{i+1}}I_{i+1}{}^{i+1}\dot{\omega}_{i+1} + \\
&\quad {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}{}^{i+1}\omega_{i+1} \quad (11)
\end{aligned}
$$

Inward Iterations: $i : L \rightarrow 1$

$$
\begin{aligned}
{}^{i}f_i &= {}^{i}R_{i+1}{}^{i+1}f_{i+1} + {}^{i}F_i \quad (12)\\
{}^{i}n_i &= {}^{i}R_{i+1}{}^{i+1}n_{i+1} + {}^{i}p_{C_i} \times {}^{i}F_i \\
&\quad + {}^{i}p_{i+1} \times {}^{i}R_{i+1}{}^{i+1}f_{i+1} + {}^{i}N_i \quad (13)
\end{aligned}
$$

where ${}^{i}p_{i+1}$ represents the position of the origin of frame i+1 in frame i, f_i and n_i represent the force and moment of link i-1 on link i, and $\hat{z}$ is a unit vector in the z-direction.

Note that the equations for the outward iterations for the link velocities and accelerations require the specification of the initial conditions for ${}^{0}\omega_0$, ${}^{0}\dot{\omega}_0$, and ${}^{0}\dot{v}_0$, which are zero for a fixed base in the absence of gravity. The equations for the inward iterations for the link interaction forces and moments require the specification of the final conditions,

$$
\begin{aligned}
{}^{L+1}f_{L+1} &= {}^{L+1}F_e \quad (14)\\
{}^{L+1}n_{L+1} &= {}^{L+1}N_e \quad (15)
\end{aligned}
$$

where ${}^{L+1}F_e$ and ${}^{L+1}N_e$ represent the forces and torques exerted on the end-effector of the robot.

The dynamics resulting from the application of the Newton-Euler Method have the general form[5]

$$M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + J(\theta)^T F_E = \tau_a \quad (16)$$

where M is the inertia matrix, c is a vector of nonlinear Coriolis and centripetal torques, F_E is the force/torque applied to the end-effector, J is the Jacobian, and τ_a is the vector of joint torques applied by the actuators. Equation (16) represents the dynamical equations of motion for the manipulator when it is attached to a fixed platform.

Coupled Dynamics

The coupled dynamics can be derived using the same Recursive Newton-Euler Method used for developing the dynamics of manipulators with fixed-bases. Instead of assuming a fixed base frame, the link 0 frame is now attached to the vehicle body frame. Although developed here for a single arm, the methodology can be extended to include multiple arms.

Equations (9), (10), and (11) can be used to write the total force and torque, ${}^{0}F_0$ and ${}^{0}N_0$, acting on the vehicle

$$
\begin{aligned}
{}^{0}F_0 &= m_0{}^{0}\dot{v}_{C_0} \quad (17)\\
{}^{0}N_0 &= {}^{C_0}I_0{}^{0}\dot{\omega}_0 + {}^{0}\dot{h}_w + \\
&\quad {}^{0}\omega_0 \times ({}^{C_0}I_0{}^{0}\omega_0 + {}^{0}h_w) \quad (18)\\
{}^{0}\dot{v}_{C_0} &= {}^{0}\dot{v}_0 + {}^{0}\dot{\omega}_0 \times {}^{0}p_{C_0} \\
&\quad + {}^{0}\omega_0 \times ({}^{0}\omega_0 \times {}^{0}p_{C_0}) \quad (19)
\end{aligned}
$$

Note that (18) is equivalent to the spacecraft attitude dynamics (4) derived in the last section. In addition, the linear and angular accelerations for link 0, ${}^{0}\dot{v}_0$ and ${}^{0}\dot{\omega}_0$, are no longer zero as they were for the fixed-base case.

The external forces and torques applied on link 0 in frame 0 coordinates are given by ${}^{0}f_0$ and ${}^{0}n_0$, respectively. These are simply the forces and torques provided by body-mounted thrusters and reaction wheel system

$$
\begin{aligned}
{}^{0}f_0 &= {}^{0}f_T \quad (20)\\
{}^{0}n_0 &= {}^{0}r_T \times {}^{0}f_T + {}^{0}\dot{h}_w \quad (21)
\end{aligned}
$$

where ${}^{0}f_T$ is the thrust vector in frame 0, ${}^{0}r_T$ is the position of the thruster in frame 0, and ${}^{0}\dot{h}_w$ is the change in angular momentum of the reaction wheel in frame 0. Just as the joint motors mounted on the arm provide the external joint torque inputs to the manipulator, so the thrusters and reaction wheel provide the external force and torque inputs to the base platform.

The dynamics can now be obtained by applying the outward recursive equations from $i = 1 \rightarrow L$ using the variable initial conditions for link 0, and then applying the inward recursive equations from $i = L \rightarrow 0$ using the terminal conditions (14) and (15). Note the following two major differences between the coupled and decoupled formulations:

- the initial conditions on the outward recursion for the link 0 velocity and acceleration, ${}^{0}\omega_0$, ${}^{0}\dot{\omega}_0$, and ${}^{0}\dot{v}_0$, are now variables rather than constants
- the inward recursion now proceeds all the way back to frame 0 in order to incorporate the effect of the external forces and torques on the base

The resulting dynamics take on a form analogous to (16)

$$
M(x_0, \widetilde{q}_0, \theta_a)\begin{bmatrix} \dot{v}_0 \\ \dot{\omega}_0 \\ \ddot{\theta}_a \end{bmatrix} + c(x_0, \widetilde{q}_0, \theta_a, v_0, \omega_0, \dot{\theta}_a) + J(x_0, \widetilde{q}_0, \theta_a)^T F_E = \begin{bmatrix} f_0 \\ n_0 \\ \tau_a \end{bmatrix} \quad (22)
$$

where the arm torque vector is

$$\tau_a = \begin{bmatrix} {}^1n_1{}^T\hat{z} \\ {}^2n_2{}^T\hat{z} \\ \vdots \\ {}^Ln_L{}^T\hat{z} \end{bmatrix} \tag{23}$$

Note that the position and velocity dependencies now encompass the entire robot state.

To a good approximation, it can be assumed that the dynamics of the reaction wheel are decoupled from the rest of the robot and are given by

$${}^CI_w\dot{\omega}_w = \tau_w \tag{24}$$

However, from (21) and (22), it is clear that the vehicle and manipulator dynamics are not decoupled from the wheel dynamics.

State Equations

The state of the robot is a combination of the vehicle, arm, and reaction wheel states. The vehicle state consists of the translational and rotational position of the arm base frame on the vehicle with respect to the inertial frame r

$$x_v \equiv \begin{bmatrix} x_0 \\ \tilde{q}_0 \\ v_0 \\ \omega_0 \end{bmatrix}, \dot{x}_v = \begin{bmatrix} v_0 \\ \dot{\tilde{q}}_0 \\ \dot{v}_0 \\ \dot{\omega}_0 \end{bmatrix} \tag{25}$$

Note that these vectors are expressed in the inertial reference frame r, but they can be easily transformed to the link 0 frame through premultiplication by ${}^rR_0$, the direction cosine matrix corresponding to the quaternion ${}^r\tilde{q}_0$. In addition, the quaternion rate and angular velocity are related via

$$\dot{\tilde{q}}_0 = \begin{bmatrix} \frac{1}{2}\omega_0 \times q_{123} + \frac{1}{2}q_4\omega_0 \\ -\frac{1}{2}\omega_0^T q_{123} \end{bmatrix} \tag{26}$$

where $\tilde{q}_0 \equiv [q_{123}^T q_4]^T$ and $q_{123} \equiv [q_1 q_2 q_3]^T$. The state vector for the manipulator arm consists of the joint angles and the joint angular rates

$$x_a \equiv \begin{bmatrix} \theta_a \\ \dot{\theta}_a \end{bmatrix}, \dot{x}_a = \begin{bmatrix} \dot{\theta}_a \\ \ddot{\theta}_a \end{bmatrix} \tag{27}$$

The state vector for the reaction wheel is the wheel angular velocity

$$x_w \equiv \omega_w, \dot{x}_w = \dot{\omega}_w \tag{28}$$

Only the velocity of the wheel is included as a state since the position of the wheel does not influence the vehicle dynamics.

The evolution of the state vector over time determines the dynamics of the system. This consists of determining the accelerations of the vehicle, $\dot{v}_0$ and $\dot{\omega}_0$, manipulator, $\ddot{\theta}_a$, and reaction wheel, $\dot{\omega}_w$, from the robot and wheel dynamics, (22) and (24), and integrating them over time.

Equation (22) represents the "forward" dynamics of the robot which describes how to determine the forces and torques applied to the robot based upon its motion. However, it is generally more useful to determine the motion of the robot based upon force inputs. Unfortunately, the "inverse" dynamics solution represented by solving (22) for the accelerations, can rarely be applied since it requires explicit knowledge of the robot inertia and Coriolis/centripetal force vector. A procedure for determining the inverse dynamics from the forward dynamics is now summarized for the case when a symbolic representation of the dynamics is not available.[6]

Assume that a numerical function of (22) exists called $DYN()$ which computes the torque based upon the robot motion using the Recursive Newton-Euler Method described earlier

$$Q = DYN(q, \dot{q}, \ddot{q}, F_E) \tag{29}$$

where Q is the generalized force representing the right side of (22), and $\ddot{q}$ is the generalized acceleration multiplying M. Then (29) can be used to determine a numerical solution for M and c using the procedure outlined below.

Coriolis/Centripetal Force Vector

The Coriolis/centripetal force vector c is a one-dimensional array of length L+6, where L is the number of degrees of freedom of the manipulator. The first six elements of c correspond to the nonlinear forces and torques in the base, and the last L elements correspond to the nonlinear torques in the arm joints. If the acceleration and end-effector force in (29) are set equal to zero,

$$DYN(q, \dot{q}, 0, 0) \rightarrow c \tag{30}$$

then the resulting output is c. Thus a stripped down version of (29) can be generated in which all the acceleration terms are removed and the input is the robot state and the "force" output is c.

Inertia Matrix

The inertia matrix is a square L+6 matrix of acceleration coefficients appearing in (22). By setting the velocities and end-effector forces in (29) to zero,

$$DYN(q, 0, \ddot{q}, 0) \rightarrow M\ddot{q} \tag{31}$$

the output becomes $M\ddot{q}$ since all of the terms in c involve products of the velocities. The jth column of M can then be found by setting the jth element of $\ddot{q}$ equal to 1

$$DYN(q, 0, e_j, 0) \rightarrow M_j \tag{32}$$

where M_j is the jth column of M, and e_j is the unit vector with the jth element set equal to unity. Similar to the case for c, a stripped down version of (29) can be generated in which all the velocity terms are removed and the input is the unit acceleration vector e_j and the force output is M_j.

While (32) represents a straightforward approach for calculating M, it is also very inefficient because it does not take advantage of the symmetry of M. Walker et al[6] describes three methods for computing M based upon a numerical solution of the forward dynamics. Method 2 was chosen because of its balance between computational efficiency and coding complexity.

Once the inertia matrix and Coriolis/centripetal force vectors have been found, the complete robot state (25)–(28) can be integrated using the accelerations derived from (22) and (24).

Attitude Stabilization

In order to stabilize the vehicle during an arm maneuver, the reaction forces and torques of the arm with the base, f_1 and n_1, must be countered. In this work, it is assumed that reaction wheels, control moment gyros, or pulse-modulated jets are mounted on the vehicle for this purpose. While it is assumed here that the torque is delivered exactly as commanded, generally, a model of the particular device should be incorporated in the dynamical model.[7]

The decoupled spacecraft attitude dynamics given by (4) can be rewritten as

$$^{C}I\dot{\omega} = u - \omega \times (^{C}I\omega + h_w) \tag{33}$$

$$u = N - \dot{h}_w \tag{34}$$

where u is the total torque applied to the vehicle by the thrusters and reaction wheels.

Let the attitude quaternion of the vehicle be given by $\tilde{q}$, and let $\tilde{q}_c$ denote the desired or commanded vehicle attitude. The quaternion error is defined as the "difference" between the desired and actual orientation and is represented by the quaternion multiplication operation

$$\tilde{q}_e \equiv \tilde{q}_c \tilde{q}^{-1} \tag{35}$$

The first three elements of the quaternion "error" represent the eigenaxis rotation between the desired and actual orientations.

The feedback controller used for this simulation is a regulator centered around the quaternion error feedback. Generally, this controller consists of three terms: linear quaternion error feedback, linear body-rate feedback, and nonlinear velocity feedforward

$$u = \omega \times (I\omega + h_w) - D\omega - K\Delta q \tag{36}$$

where Δq is a 3x1 vector consisting of the first three elements of $\tilde{q}_e$, and ω is the rotational rate of the body in the body frame. D and K are constant gain 3x3 matrices which depend on the desired bandwidth and damping ratio of the controller.

The torque control law given by (36) is achieved by a combination of thruster and reaction wheel inputs as determined by (21). The gyroscopic feedforward term is used to counteract the spacecraft cross coupling but is often eliminated unless the rates are very high. The rate feedback damps the attitude about zero velocity. The quaternion error term reduces the size of the orientation error through negative feedback. If K and D are chosen to be scalar multiples of the spacecraft inertia, i.e. $K = kJ$ and $D = dJ$, then it can be shown that this control law achieves the desired eigenaxis rotation.[8] The closed loop dynamics in this case can be found by substituting the control into the spacecraft dynamics yielding

$$\dot{\omega} = -d\omega - k\Delta q \tag{37}$$

For simplicity, K and D were chosen to be diagonal with elements given by $k_{ii} = \omega_n^2 I_{ii}$ and $d_{ii} = 2\zeta\omega_n I_{ii}$, respectively, where I_{ii} is the principal inertia of the ith body axis in frame C.

Control Station

Operators are able to view a blend of video, graphical simulation, and control station panels as shown in Figure 4. Simulated views from external and onboard vehicle cameras are displayed. Stereo monitors are provided to allow operators to view stereo video from the vehicle using CrystalEyes^{TM} LCD glasses. As shown in Figure 5, the graphical simulation allows users to fly around the virtual environment providing views from many angles which live video may not be able to provide.

Many different computer processes are required to control the Ranger vehicle. These processes are broken down into groups according to the functions they perform and often exist in several locations. The I/O device modules and control station modules are located on different platforms across the network. The onboard controllers and DMS reside on the Ranger main computer. Both the actual vehicle and the simulation use the same I/O and control station modules.



Fig. 4 Operator using control station.

The most prominent use of command and telemetry occurs in the control station modules. These modules act as the interpreter process, consuming device data and generating vehicle commands. The telemetry is displayed across several control panels which can also be used to send commands to the vehicle. Input to the control station modules from hand controllers, forceballs, and position trackers are interpreted by the module, which then generates commands for the vehicle.

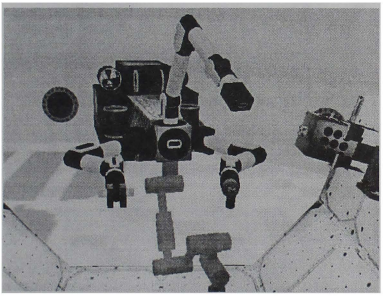


Fig. 5 Graphical model of Ranger NBV.

This process of using distributed modules instead of a centralized module for control is called "federated distributed control".[9] The control modules do not need to be colocated and can exist on different machines, platforms, or even on separate local networks which are interconnected across the internet. Interprocess communication using Network Data Delivery Service (NDDS) by Real Time Innovations, Inc., enables effective implementation of this federated distributed system.

The manipulators and vehicle can be controlled by many different control devices including 3-axis hand controllers, forceballs, head trackers, and a 6-dof mouse. Each control station module can use any active input device. This enables one input device to drive several arms and the vehicle simultaneously. The modularity of the system allows multiple copies of the same control station module to be run in parallel. For example, in Figure 6, a left dexterous control station module is active on two different machines, CS 1 and CS 2. While both are updated with telemetry, the command checker allows only one to send commands to the simulated vehicle.

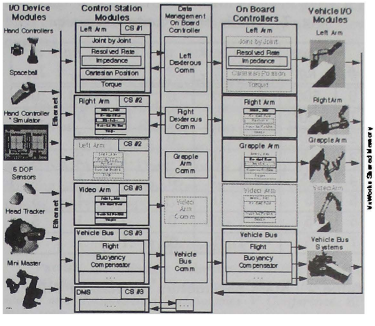


Fig. 6 Control Station Modules.

Simulation Results

A real-time simulation of the vehicle dynamics with arm coupling were written in "C" on an SGI platform. The position of the tool tip and the orientation of the tool were controlled by a pair of 3-dof hand controllers illustrated in Figure 7. The Denavit-Hartenburg parameters for the manipulator are given in Table 1. The position of the tool tip from the center of the wrist was 0.508 m. The bandwidth of the joint servo controllers was 5 Hz and the damping ratio was 1.

link	$a_{i-1}(m)$	$\alpha_{i-1}(rad)$	$d_i(m)$	$\theta_i(rad)$
1	0	0	0	θ_1
2	0	$-\pi/2$	0	θ_2
3	0	$\pi/2$	0.508	θ_3
4	0	$-\pi/2$	0	θ_4
5	0.152	$\pi/2$	0.394	θ_5
6	0	$-\pi/2$	0	θ_6
7	0	$\pi/2$	0	θ_7

Table 1 D-H parameters for Ranger arm.

Table 2 shows the mass, center of mass, and principal inertias used for the vehicle (link 0) and the arm

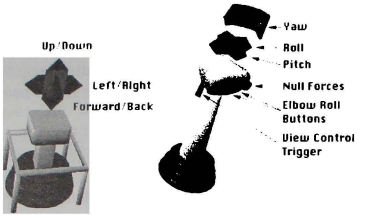


Fig. 7 3-axis hand controller pair.

#	mass kg	mass center, m $\bar{x}$	$\bar{y}$	$\bar{z}$	inertia, $kg-m^2$ I_{xx}	I_{yy}	I_{zz}
0	500	0	0	0	100	100	100
1	20	0	0	0	0	0	20
2	20	0	-.2	0	.5	.1	20
3	15	0	0	-.1	.2	.2	10
4	15	.1	-.2	0	.5	.5	10
5	5	0	0	-.1	.1	.1	5
6	5	0	-.05	0	.2	.1	5
7	5	-.1	0	.05	.1	.1	5

Table 2 Mass parameters for Ranger NBV.

in the simulations. The only nonzero cross-product of inertia was $I_{4_{xy}} = 2kg-m^2$ due to the asymmetry of the elbow link. The vehicle mass was roughly five times the total mass of the arm. However, the inertia of the arm when fully extended was significant when compared with the inertia of the vehicle. The reaction wheel inertias were $1kg-m^2$ about each of the principal axes.

Simulations were first performed in "free-floating" mode where the vehicle thrusters were diabled. Figure 8 and Figure 9 show a timed exposure of the motion exhibited by the vehicle during a typical side-to-side and up/down motion of the end-effector. The reaction forces are so severe in both cases that the motion of the arm manifests itself as more of a yaw and pitch motion of the wrist rather than side-to-side and up/down motion of the end-effector, respectively.

The same arm commands were then tested with the attitude stabilization turned on and the control bandwidth of the attitude control system set to 1 Hz. The vehicle attitude remained very stable as shown Figure 10 and Figure 11. While the vehicle did translate as the arms moved, the arms exhibited the desired straight line motion. The predominant torques exerted by the reaction wheels in both cases are shown in Figure 12 which were roughly in the 3-4 N range when the tool tip was coasting at 1/4 m/s.

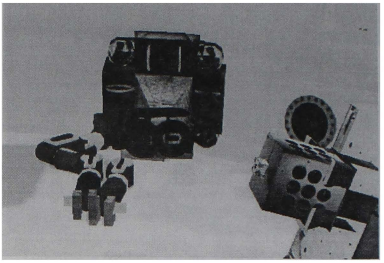


Fig. 8 Side command in free-floating mode.

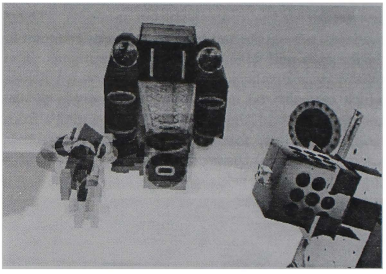


Fig. 9 Up command in free-floating mode.

These may be pushing the limits of most commercial reaction wheels, but it is well within the range of alternative torquer devices such as control moment gyros.[10]

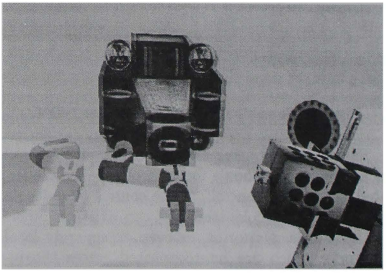


Fig. 10 Side command with ACS turned on.

Conclusion

The main purpose of this paper was to show how to incorporate a free-flying base in the standard It-

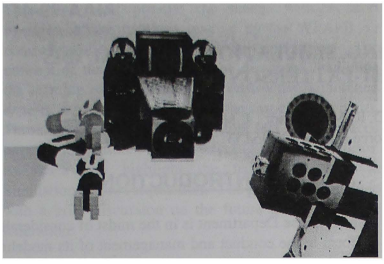
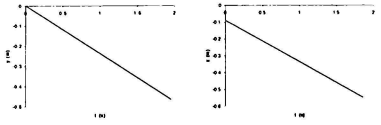
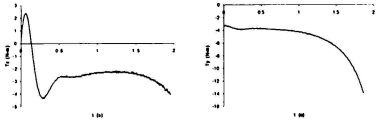


Fig. 11 Up command with ACS turned on.



a) Side displacement. **b) Up displacement.**



c) Yaw torque. **d) Pitch torque.**

Fig. 12 Reaction wheel torques for yaw and pitch maneuvers.

erative Newton-Euler formulation of the dynamics. An inverse dynamics procedure was outlined to implement a numerical solution of the resulting coupled forward dynamics. The inverse dynamics were then integrated into a graphical simulation for use as a training simulator for implementing tasks. Commands to the arm were input through a pair of 3-axis hand controllers to control the translation and rotation of the end-effector.

Uncompensated arm reaction forces produced substantial movement of the vehicle making it virtually impossible to achieve straight line motion of the arm. An attitude control scheme was then implemented to reduce the vehicle motion caused by the arm reaction torques. While significant translation still took place, the rotation of the vehicle was sufficiently reduced to make arm tasks much more manageable. The ability to achieve the commanded straight line motion of the end-effector was key in this result.

Although the development of an attitude stabilization scheme was not a focus of this research, it is encouraging that even a pure feedback regulator was able to effectively counter the reaction torques from the arm. This was the case for even relatively low bandwidths of 1 Hz. Recent tests incorporating an adaptive formulation of the quaternion feedback regulator indicate that still further improvements are possible.[11,12] The reaction torques observed in these simulations indicate capabilities well within the range of commercial control moment gyros if not high capacity reaction wheels currently under development.

Acknowledgements

The authors would like to acknowledge the many people on "Team Ranger" who made Ranger NBV a reality. In addition, we would also like to acknowledge NASA and Dave Lavery of HQ for his continuing support of the Ranger project under NASA Grant NAGW-2245.

References

[1]Parrish, J., "Ranger Telerobotic Shuttle Experiment (RTSX): Status Report," *Proceedings of the SPIE Telemanipulator and Telepresence Technologies V Conference*. SPIE, Nov. 1998.

[2]Graves, J., "Ranger Telerobotic Flight Experiment Program Update," *Proceedings of the SSI/Princeton Space Manufacturing Conference*, Princeton, May 1995.

[3]Kaplan, M., *Modern Spacecraft Dynamics and Control*, John Wiley and Sons, New York, 1976.

[4]Craig, J., *Introduction to Robotics: Mechanics and Control, 2nd ed.*, Addison-Wesley, Reading, Mass., 1989.

[5]Lewis, F., Abdallah, C., and Dawson, D., *Control of Robot Manipulators*, MacMillan, New York, 1993.

[6]Walker, M. and Orin, D., "Efficient Dynamic Computer Simulation of Robot Mechanisms," *J. of Dynamic Systems, Measurement, and Control*, Sept. 1982. pp. 205–211.

[7]Wie, B. and Barba, P., "Quaternion Feedback for Spacecraft Large Angle Maneuvers," *J. of Guidance, Control, and Dynamics*, May 1985, pp. 360–365.

[8]Wie, B., Weiss, H., and Arapostathis, A., "Quaternion Feedback Regulator for Spacecraft Eigenaxis Rotations," *J. of Guidance, Control, and Dynamics*, May 1989, pp. 375–380.

[9]Lane, J. C., Carignan, C., and Akin, D., "Reconfigurable Control Station Design for Robotic Operations," *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, IEEE, Oct. 1997.

[10]Wertz, J., *Spacecraft Attitude Determination and Control*, Kluwer Academic, Norwell, MA, 1978.

[11]Graves, J., *Design and Control of a Vehicle for Neutral Buoyancy Simulation of Space Operations*, University of Maryland, M.S. Thesis, College Park, 1997.

[12]Henshaw, C. G., *Nonlinear and Adaptive Control Strategies for Improving the Performance of Teleoperated Spacecraft*, University of Maryland, M.S. Thesis. College Park, 1998.

THE "NEW" AIR COMBAT MODELING, SIMULATION, AND ANALYSIS FEDERATION AT WRIGHT-PATTERSON AFB, OH

S. M. Rodgers*, G. K. Hellmann†, T. E. Menke‡, D. R. Mittlesteadt*, J. M. Zeh†, and D. K. Richart‡
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

Sweeping changes throughout the Department of Defense to improve efficiencies in acquisition processes have led to the creation of the simulation and analysis facility, or SIMAF, at Wright-Patterson AFB (WPAFB), OH. SIMAF is owned and operated by the Aeronautical Systems Center (ASC) in a substantial teaming partnership with the Air Force Research Laboratory (AFRL). The facility is staffed with a new modeling, simulation, and analysis federation. The federation has been formed through partnerships with engineering, analysis, and simulation offices at WPAFB. With SIMAF, this federation is conducting simulation and analysis for system and requirements trades in a way that is prototypical of the Simulation Based Acquisition vision. SIMAF fulfills the need for a facility appropriate for conducting highly classified simulations. This capability enables collaborative teams of USAF decision-makers, warfighters, and analysts to better and more quickly evaluate and develop advanced concepts and technologies for military applications. The federation's innovations include a more rigorous analytical approach to modeling and simulation using state-of-the-art distributed computing capabilities. The analysis approach capitalizes upon the complementary strengths of constructive and virtual simulation methodologies. SIMAF provides cutting-edge visualization capabilities, computing power, and air-to-air and air-to-ground combat simulations.

INTRODUCTION

The Defense Department is in the midst of considerable reform in the conduct and management of its modeling and simulation activities. There are a large number of new initiatives and streamlining measures.

Jacques S. Gansler, the Under Secretary of Defense for Acquisition and Technology, noted in a 1998 address to the National Defense Industrial Association's Simulation Based Acquisition Workshop that one of the top challenges for the department was to modernize the military with drastically reduced cycle times and lower cost[1]. Inherent within the vision for Simulation Based Acquisition is that advances in the use of modeling and simulation technologies are key elements to meeting these challenges.

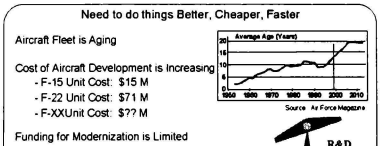


Figure 1 Aging Defense Systems and Limited Funding

The June 1995 4-star summit provided a New Vector for Air Force modeling and simulation[2]. This vision was endorsed by Air Force senior leadership and called for eliminating redundancy within the Air Force's inventory of Modeling and Simulation (M&S) tools. The proposed consolidation should directly lower operations and maintenance costs associated with M&S by reducing multiple model upgrades and duplicative database maintenance[3]. Many in the community thought that increasing the collaboration between existing users of models and simulation would speed the migration toward common models and databases.

* Air Combat SPO, Aeronautical Systems Center
† Integration and Assessment Branch, Air Force Research Laboratory
‡ Virtual Simulation Group, Aeronautical Systems Center
This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

This paper will provide the reader with a short overview of the facilities making up the WPAFB air combat MS&A federation and its capabilities. The concept of the federation will be discussed, including the teaming of the virtual and constructive simulation experts, why it is necessary, and the associated benefits. Throughout the discussion, the emphasis will be placed on the federation's recently completed simulations. Examples will highlight the federation's role in Simulation Based Acquisition. The paper will conclude with a brief discussion on the future direction of the federation.

BACKGROUND

Aeronautical Systems Center (ASC), as an acquisition management organization, is tasked with the development of Air Force systems. Inherent within this management duty is a number of tasks including:

- technical analysis of user requirements;
- dissemination of new technologies;
- applications of these technologies to Air Force requirements developers, systems prototype developers, and formal system program office (SPO) development organizations;
- developmental testing;
- transition of new systems to operational user testing and evaluation; and
- transition to operational logistics support.

In addition to the SPOs and planning offices within ASC, a number of directorates within the Air Force Research Laboratory and the National Air Intelligence Center (NAIC) are also hosted at WPAFB. There were opportunities for synergies among these ASC planning organizations, laboratories, intelligence offices and system program offices. Exploited correctly, these synergies would result in more effective and efficient development of systems that clearly satisfy the warfighter's needs.

To answer the challenges of reform and streamlining and to capitalize on the synergistic opportunities offered by the expertise present at WPAFB, ASC developed the simulation and analysis facility, or SIMAF. SIMAF is an important enabler for ASC to sustain itself as the Defense Department's air vehicle center of excellence for distributed modeling and simulation for systems development. To support this mission area, a Congressional insert of $7M was included in the FY97 Omnibus Appropriations Bill (S11936/HR4278/HR3610) for a "WPAFB Simulation Facility" as part of PE27601F, "USAF Wargaming and Simulation." The objective was to significantly improve ASC's M&S infrastructure and to serve as ASC's entree into the Joint Synthetic Battlespace[4].

One immediate need was a facility appropriate for conducting highly classified simulations across a variety of programs. SIMAF would provide the environment for interdisciplinary virtual prototyping and testing. At this facility analysts, warfighters, and USAF decision-makers could collaborate to:

1) speed identification of promising technologies and concepts;
2) perform evaluations within a "system of systems" framework; and
3) add additional rigor to the acquisition process.

During the early development of SIMAF, the existing simulation facilities at WPAFB provided significant insight and perspective for SIMAF's definition, including required hardware, software, physical layout, and network connections. It was during this planning period that a new relationship between the AFRL Integration and Assessment branch's virtual simulation experts and the ASC Planning Directorate's analysts was formed. The members of these teams knew that the final objective was to use advanced simulation to identify, quantify, and develop promising technologies for the warfighter.

SIMAF was to be ideally situated among other existing simulation facilities at WPAFB to leverage existing capabilities and expertise. ARFL's Collaborative Simulation Technology Branch (AFRL/IFSD) in the Sensors Directorate was one of two sites for SIMAF's initial operational capability. The other was in the Air Vehicle Directorate's Integration and Assessment Branch (AFRL/VACD). Beyond these, there were numerous other offices that added MS&A expertise to the landscape at WPAFB. These included the Human Effectiveness Directorate's Pilot-Vehicle Integration Laboratory (AFRL/HECI) and in ASC, the Engineering Directorate has had the Crew Systems Evaluation Facility (ASC/ENFC) in the same simulation complex since 1996. Also, the Planning Directorate (ASC/XR) has had analysis groups specializing in both constructive and virtual simulation. Within NAIC, the Engagement Analysis Branch in the Directorate of Technical Assessments has a wealth of experience in threat modeling for threat assessments. The vision for work in SIMAF then created an organizational bridge coupling the resources within these various organizations. This arrangement fostered a new air combat modeling, simulation, and analysis (MS&A) federation. This "new" federation was fortified through

teaming partnerships that shared the facilities' resources, both computer and human.

WPAFB FEDERATION

Prior to this teaming with ASC's Planning Directorate, assessments of the technologies and concepts done within the Integration and Assessment Branch had not been focused at the mission or campaign level. This changed starting with the recent VISTA Advanced Capabilities Study, which looked at aircraft agility through thrust vectoring, missile agility through off-boresight target track capability, and a helmet-mounted display and targeting system. The Planning Directorate's analysts investigated what impact, if any, these virtual results had at the mission and campaign level through the use of constructive analysis. This activity fanned the smoldering idea that jointly, Integration and Assessment Branch virtual simulation engineers and Planning Directorate analysts could form a potent air combat MS&A team. As collaboration increased this idea burned brighter. Soon it was apparent that SIMAF was to be much more than a new "*physical*" facility. Rather, its unique worth was as a "*people*" facility. This team of virtual simulation scientists, engineers, and analysts would be the heartbeat of SIMAF's success. Specifically, this team would bring to bear a superior analytical approach and a broad interdisciplinary composition.

Analysis Approach

In the past, most laboratory technologies and concepts had been evaluated in isolation—at the component or sub-system level. Rarely were technology assessments taken up to the mission level and rarer still to the campaign level. Of course, it is precisely at these higher levels that innovative technologies and new concepts should clearly demonstrate their military worth to secure development backing and funding.

Simulation Based Acquisition requires the application of the appropriate models to answer the questions being asked by the decision-makers. Simulations and exercises without analytic underpinnings designed to address these specific questions fail to realize the benefits that are sought under the Simulation Based Acquisition initiative. The results that come from the application of the considerable computational power available today can be quite convincing when based on sound principles and systems engineering methods. This requires a broad interdisciplinary team composed from the scientific and engineering fields that are germane to the systems being studied. In this application, virtual and constructive analytical methods substantively complement each other.

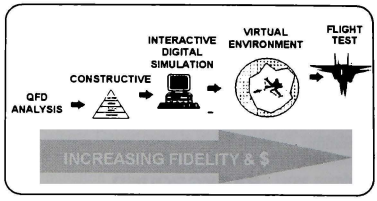


Figure 2 Simulation Spectrum

In simulation, the term "constructive" refers to simulations that execute without direct human interaction during runtime. These simulations typically are run faster than real time with hundreds or thousands of repetitions. The main strength of constructive analysis is the ability to achieve statistical significance in the study results. These results are needed to validate the study's assumptions and to aggregate higher fidelity results to the next level of analysis. For example, constructive engagement-level results are aggregated for input to mission-level or campaign-level studies. Then, these constructive results are used as inputs and assumptions in virtual studies. Although Figure 2 shows this as a sequential process, there is considerable iteration. Results from the various studies flow both up and down the spectrum. Another strength of constructive studies is the ability to accomplish sensitivity analyses for given attributes of the system under test. These sensitivity and other constructive studies are well suited for screening efforts for the virtual studies.

The term "virtual" refers to simulations that run at near real time with direct human interaction through a participant within the simulation. In this regard, virtual simulation includes low to high fidelity piloted simulators. The primary strength of "virtual" is the increased realism and validation of the tactics applied. Without a human in the loop, realism is especially hard to capture with respect to the free-flowing interactions of elements within the synthetic warfare environment. For air combat tactics, the pilot serves as the domain expert. When immersed within the synthetic environment, the pilot can observe these interactions, respond according, and provide valid tactics that are optimized for the system and the scenario. A military

system, like any tool, can be employed correctly or incorrectly. The correct tactical employment of a military system will be optimized for a given scenario and will have a first order impact on the outcome of scenario.

This increased realism in tactics also provides excellent insight into pilot situation awareness (SA), pilot-vehicle interface issues, decision making, and execution timeline issues. Situation awareness is further decomposed to encompass information display and processing issues, target and threat classification, discrimination, and identification (ID) issues, subjective issues, and more. These strengths are highly complementary in an integrated analytical plan and the "design of experiments" process[5].

The end benefit of placing pilots in the cockpit and collecting analytic data is substantial. The data are used to compare trends between the constructive and virtual efforts. Validated tactics are then inserted into the constructive model. Pilot qualitative assessments are also incorporated into the simulation where practical. The tactics and assessments provide a direct link between the constructive "pilot-less" environment and the virtual "piloted" environment. The virtual analysis effort does not replace the constructive effort; rather, it is complimentary in nature.

Virtual efforts are both expensive and time consuming. As such, a virtual effort can rarely satisfy the rigorous constraints of probabilistic analyses. By incorporating information and data from virtual simulation into the constructive process, the analyst can produce statistically significant results that are based on valid tactics. These validated tactics are a direct result of the increased scrutiny of domain experts and their interaction with the simulation.

In addition to mainstream analytical benefits of virtual simulation, there are other useful effects from putting the operator "in the loop." One benefit that should be obvious but is at times overlooked by analytical purists is the persuasive nature of taking part in the event. Many decision-makers have a healthy skepticism concerning study results. In overcoming this skepticism there are few substitutes for getting the user involved in the details and nuances that are seen first-hand during participation.

Correlating constructive and virtual simulation is an iterative process. Pre-virtual constructive analysis can be used not only for statistical purposes but to screen the more expensive and time intensive virtual experiments. The goal of the prescreening effort is to identify system attributes worthy of more detailed piloted evaluation. This pre-screening effort feeds the experiment matrix for the virtual events. The analysis results of the virtual events are later correlated with the pre-virtual constructive analysis for significant trends in the various measures of effectiveness (MOEs). The results of the virtual simulation also supports post-virtual constructive analysis. This constructive analysis begins with an experiment matrix based in large part on the results of the virtual simulation. The virtual simulation's MOEs, situation awareness, pilot vehicle interface, target classification, target recognition, target discrimination, target identification, and tactics are used by various engineering disciplines to correlate constructive analysis with piloted simulation results. The tactical refinement identified in the virtual events improves the statistical results obtained from the follow-on constructive analysis efforts. Often, virtual simulation provides unique and innovative insight in the employment of systems previously not conceived of by the engineering, test, or warfighter communities. Often particular attributes of interest become more important to objectively quantify following the virtual events.

This analysis process is iterative in that pre-virtual, virtual, and post-virtual analysis results focus the next modeling, simulation, and analysis cycle. This process is illustrated in Figure 3.

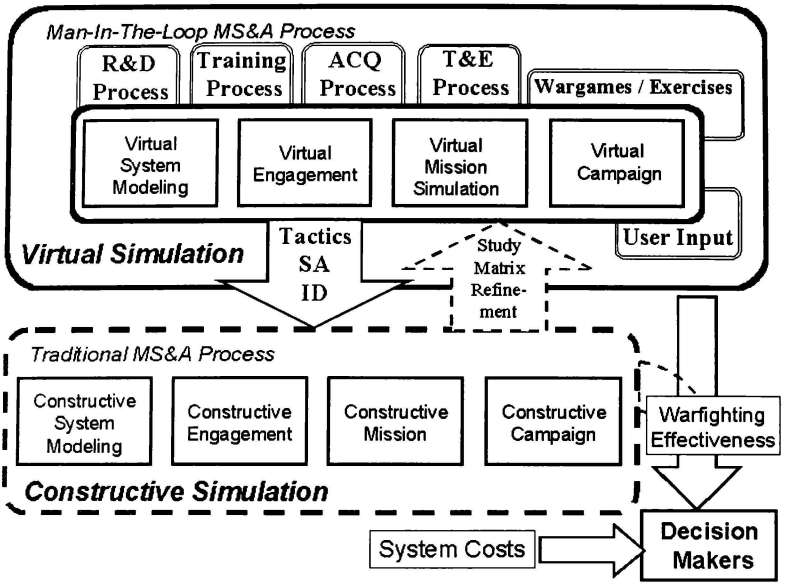


Figure 3. Constructive and Virtual Simulation Complement Each Other

The Engineering and Analysis Team

The engineering and analysis team comprises expertise from a number of diverse disciplines including: project management; software development; hardware integration; graphics development; aerodynamic performance, guidance, navigation, and control; threat modeling; database management; aircraft design; avionics design; mission planning; and analysis. The core WPAFB team is comprised of approximately 35 professionals representing the system program offices, engineering, laboratory, and intelligence organizations. This team is equally populated by both government and contractor members.

This team begins event planning by software and hardware engineers and analysts meeting with the system program office and warfighters to address pressing questions in support of developing an operational requirements document. The analysis community scopes the attributes desired by the warfighters and system program office by iterating a development schedule with the software engineering and hardware team. The final product of this process is a detailed and highly integrated test matrix based on the design of experiments. The matrix is carefully screened to include blocking techniques to minimize individual pilot biasing effects on the resulting MOE trends while providing each pilot group with maximum exposure to each test attribute. The test matrix is designed to reach a "level 4" matrix resulting in primary test attributes that are independent.

The collaborative events done by this federation have become a prototype for SIMAF's SBA concept. The WPAFB air combat MS&A federation is uniquely qualified to provide the front end of the SBA approach and is positioning itself to complete the simulation based acquisition circle with the flight test and training communities.

SIMULATION AND ANALYSIS FACILITY

During the early development of SIMAF, the existing simulation facilities provided significant input into SIMAF definition, including required hardware, software, physical layout, and network connections[4]. SIMAF would need presentation facilities for 100 people (see Figure 4), floor space of over 40,000 square feet, segregated areas for classified work, connectivity, analyst workstations, and piloted simulators.

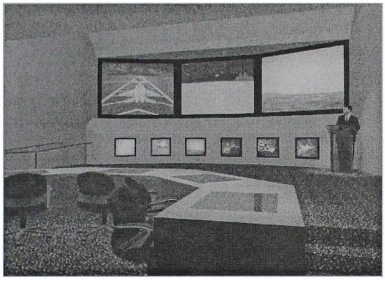


Figure 4 SIMAF Main Video Wall

In the collaborative battle rooms, two-sided virtual, wargaming and analysis are done at the campaign level. SIMAF accommodates the Thunder model, in both real-time and compressed modes. Two team areas are provided (see Figure 5); each team area accommodates eight analysts with the ability to isolate the two areas from each other. Team areas have flexible computer processor and mass storage access, visual line of sight and direct audio contact with the video wall and simulation director areas (see Figure 4). Each team area has a commander's station that includes a computer generated map table. The map table is used for monitoring analyst workstation displays, stealth viewing, moving map displays, and mission planning displays. In addition to stealth viewing, each map table has a capability for recording and playback.

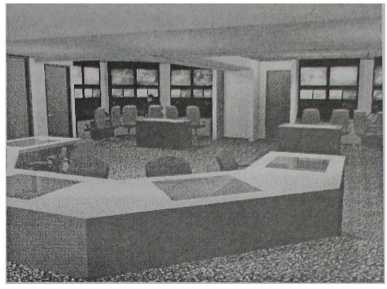


Figure 5 Director Control and Team Areas

Connectivity to several WPAFB assets have been demonstrated including the Major Shared Resource Center and the Avionics Engineering Division in ASC and the Collaborative Simulation Technology Branch in AFRL. Connections to external assets (outside of WPAFB) have also been demonstrated. These include hardware-in-the-loop and installed system test facilities at the Air Armament Center, Eglin AFB; the Decision Support Laboratory at Space and Missile Center, Los Angeles AFB; and the Modeling and Simulation Center at Electronic Systems Center, Hanscom AFB. The Bennefield Anechoic Facility at Air Force Flight Test Center, Edwards AFB and the Air Combat Environment Test and Evaluation Facility (ACETEF) at the US Navy's Patuxent River Naval Air Station are planned locations for distributed simulations within the first year of operation. Other general wide-band connections exist and are being used in 1999 to participate in JEFX99 that included live, airborne test assets.

ASC had no general-purpose facility that could accommodate the spectrum of classification levels. Additionally, there had been limited capability for multiple, distributed real-time simulation coordination and control. SIMAF has resolved this shortfall in a big way. Up to six simultaneous yet isolated highly classified simulation events can be conducted in SIMAF. Any combination of connections between these events can also be realized as required by the various programs. Both air-to-air and air-to-ground simulation exercises were conducted for the Joint Strike Fighter Program.

Piloted Air-to-Air Simulator

For air-to-air combat simulation the facility contains eight pilot stations, two command and control stations,

and one simulation director console. Each pilot station is powered by a Silicon Graphics O2 while the simulation control station is on a 2-processor Silicon Graphics Octane (see Figure 6). The two sides (red and blue) of the scenario are separated to limit cheating. Each side has intra-flight voice communication including the command and control station. Through the command and control station, input from air-based weapons controllers, ground-based intercept controllers, or other intelligence and reconnaissance sources can be injected. This enables scenarios that include a broader "system of systems."

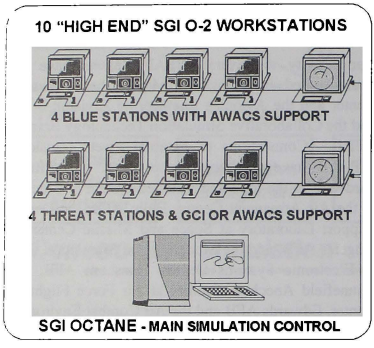


Figure 6 SIMAF Piloted Air Combat Layout

The main simulation engine is MIL-AASPEM (Man in the Loop - Air to Air System Performance Evaluation Model) II, or MIL2. MIL2 is a real time, interactive, many-versus-many model to calculate the engagement of multiple platforms. It is used to conduct fighter weapons system effectiveness analyses, tactics development and requirements analyses. MIL2 is used primarily for air-to-air analyses, but has some surface-to-air and air-to-ground capability. Aircraft in the simulation can be flown either with the piloted stations or can be fully computer driven. The simulation can also run in a batch mode to evaluate numerous scenarios, decision matrices, and a variety of factors of interest.

The main simulation consists of an object oriented pseudo-real-time programmable cyclic executive and works through shared memory and networking. It takes advantage of multiple processor machines to increase the number of real time players. These players are run in separate processes on the main simulation node. The main simulation also works in a director mode to control all of the separate processes and in a viewer mode to provide an overview of the battle space (see Figure 7). All tools also can be brought up in an object browser mode to browse the internal object hierarchy. This includes a rule base browser to interrogate the rule bases.

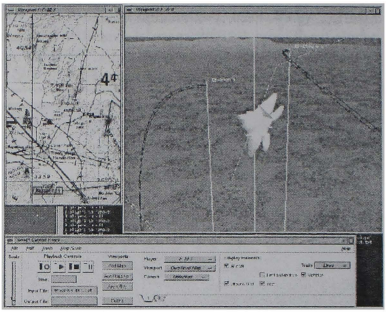


Figure 7 MIL2 Controller Display

The pilot station includes a hands-on stick and throttle (HOTAS) for pilot input and an OpenGL based pilot cockpit and out-the-window display compete with HUD (see Figure 8).

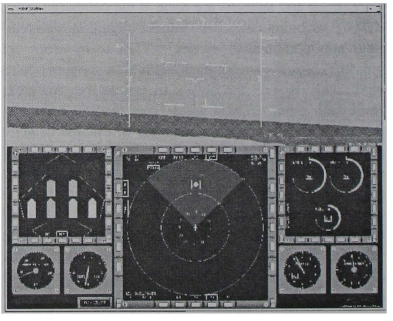


Figure 8 MIL2 Pilot Station Display

Maneuver logic is performed for each ship using a deterministic rule-based engine. The Pilot Decision Logic process performs the global decision (or command) logic for group tactics (see Figure 9).

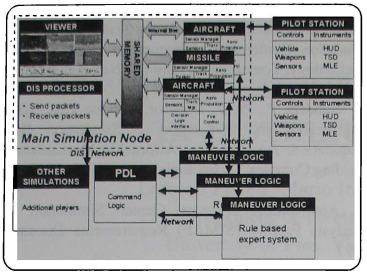


Figure 9 MIL2 Architecture

MIL-AASPEM's main simulation is networked to all of the other processes distributed across a network through programmable packets. In addition it can be connected to a Distributed Interactive Simulation (DIS) network for interaction with other DIS simulations or it can interact with the joint interim mission model (JIMM) through it's shared memory interface. In the future, MIL-AASPEM will be compatible with the High Level Architecture (HLA).

Piloted Air-to-Ground Simulator

Strike, Suppression of Enemy Air Defenses, and other air-to-ground missions are performed in the Synthetic Warfare Collaborative Environment, or SWCE. SWCE consists of the fighter requirements evaluation demonstrator (FRED) cockpit and JIMM as the environment and interactions simulation engine. FRED is made up of several software modules, each supporting a different area of the simulation. The environment modules manage the position, orientation, and velocity of the ownship, based on inputs from the flight control system. These modules also manage the movement and weapons deployment of all non-ownship entities. They also manage the flight of all weapons that have been launched by the ownship or by a threat. The sensor modules provide data to simulate on-board sensors including an infrared search and track, electronic support measures, identification friend or foe, missile warning, datalink, and targeting sensors.

The mission computer module serves as the core avionics system for FRED. It provides navigation data, and manages various components of the simulation based on pilot input including, route management, weapons stores management, tactical sensor management (e.g., sensor modes and fields-of-view), automatic modes management, defensive reaction module management, and sensor fusion management.

The advanced information management system component utilizes the Joint On-board/Off-board sensor fusion software to fuse on-board and off-board sensor tracks. This component filters out unwanted sensor reports and fusion track files to reduce fusion processing requirements and cockpit display clutter.

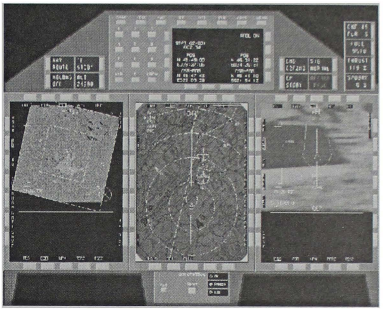


Figure 10 Generic Strike Fighter HDD

The data from the various modules within FRED are displayed on the head-down displays (HDD) and the head-up display (HUD). In SIMAF, the HDD is on a 29" monitor while the "out the window" view and HUD are projected on a 52" screen. The HDD includes an up-front controller and three multi-function displays (MFD). The MFDs accommodate a number of formats including the air-to-ground and air-to-air radar formats, a tactical situation display, an infrared targeting system format, an aircraft management system format, and electronic flight instruments. Figure 10 shows the radar format, tactical situation display, and the infrared targeting system on the three MFDs. These displays are touch sensitive so that the buttons located around the perimeter of the MFDs provide pilot control similar to an actual cockpit. The up-front controls are touch sensitive areas to simulate pushbuttons, a keypad area, and a data entry display.

The Road Ahead

Future efforts are focused to address mission and campaign-level run-time issues with new meta-modeling approaches using response surface based

techniques. These approaches hold promise to increase the fidelity of the simulations without sacrificing run-time. Particularly interesting are the areas of high dimensionality due to interaction effects within the problem domain.

Additional theater assets are being added to the strike mission scenarios including developmental weapons and advanced command and control concepts. New approaches for modeling sensors are being explored and will be tested during 1999 and 2000. Further efforts to ensure the simulation environment is suitable for test agencies include the generation of multi-spectral databases for the western test ranges.

SUMMARY

Many agencies within the Department of Defense are focused on exploiting the continuing improvements in simulation technologies. The many initiatives to reform and streamline research, development, and acquisition have lead to the Simulation Based Acquisition initiative. At WPAFB, several diverse but related offices have allied to form a new air combat modeling, simulation, and analysis federation. This federation is capitalizing on their human and computer resources to reinforce the application of solid systems engineering and analytical processes upon early acquisition efforts for military modernization.

The new SIMAF facility has enabled considerable improvement in highly classified analysis of military products in a "system-of-systems" framework. The strengths of virtual and constructive analysis have complemented each other well within this environment.

The collaborative events done by this federation have become a prototype for SIMAF's Simulation Based Acquisition concept. The federation is successfully applying Simulation Based Acquisition methods for requirements and concept development and is advancing the simulation environment to strengthen its suitability for the flight test and training communities.

REFERENCES

1. Gansler, J. S. Under Secretary of Defense Acquisition and Technology, "Life-Cycle Modeling and Simulation A Key Element In Acquisition Reform," National Defense Industrial Association, Simulation-Based Acquisition Workshop, Orlando, Florida, March 17, 1998.

2. Moorman, T. S., Jr., General, USAF, Keynote speech to the Interservice/Industry Training System and Education Conference, Albuquerque, N.M., "Today's Technology, Tomorrow's Training," Nov. 14, 1995.

3. Sanders, P., Ph.D., Director, Defense Test, System Engineering and Evaluation, "Simulation Based Acquisition: An Effective, Affordable Mechanism for Fielding Complex Technologies," March 21, 1997.

4. Aeronautical Systems Center Simulation and Analysis Facility (SIMAF) Requirements Document, 1997

5. Menke, T. E., Briefing to the Defense Planning and Analysis Society, "The Use of Virtual Simulation in Support of the Joint Strike Fighter's Joint Operational Requirements Document (JORD): A practical Simulation Based Acquisition (SBA) example," April 14, 1999.

LOW COST FAST TRAINER PROTOTYPING VIA DISTRIBUTED SIMULATION

Lorenzo Pollini* Mario Innocenti† and Elisabetta Ronchieri ‡
Department of Electrical Systems and Automation,
University of Pisa, Via Diotisalvi 2, 56126 Pisa, Italy

ABSTRACT

Developing an operator trainer for a real or a not yet existing plant or vehicle is a need in most engineering fields. The in-house development process may be very long if the necessary technological tools are not available. This paper proposes a low cost tool to develop in-house immersive simulators/trainers for a wide range of plants using the same tools system designers use in the plant design phase. The problem addressed is the capability to develop a trainer using heterogeneous commercial simulation software (i.e. Matlab and Simulink) together with custom written simulator components language/operating systems independent. The same tools used in unmanned simulations can be quickly converted to a trainer component and used in a "quasi" Real-Time fashion.

INTRODUCTION

The main goal of this work is to achieve high flexibility of simulation environment and modularity/reusability of simulator/trainer components showing that the basic skills needed to build a simple trainer prototype can be reduced to visually build system parts with tools like Simulink and connect them together and with I/O devices like real cockpits and synthetic environments visualization software. The adoption of a common network interface: the Internet Protocol (IP), that is operating system independent, allows then mixing heterogeneous simulator components. Such a simulation environment is easy and fast to setup; since communications are based on TCP/IP or UDP/IP, the only constraint on the single simulator component is hardware support for these protocols. Hence isolate Simulink schemes can be easily transformed into parts of a distributed simulator. Real Time requirements, in a first approximation, can be transformed into speed requirements since every computing unit has a single task (excluding operating system daemons), and real time synchronization can be performed with local RT clocks or via the intercommunication system.

TRAINER PROTOTYPING

The main components of a trainer are the plant dynamics simulator, the visual display system, input console and supervisor console. The first three components are strictly dependent because their interaction has stringent timing requirements. The supervisor console instead, has looser synchronization needs and can work, at worst, asynchronously. Complex dynamics simulation and synthetic environment creation are high computational load tasks; the former performs numerical integration of differential equation and exploits the floating point unit. Creation of a synthetic environment involves computer graphics imagery and needs 3D graphics boards; while most of geometric and texturing pro-

*PhD Student.
†Associate Professor, Asociate Fellow AIAA.
‡Graduate Student.
Copyright ©1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

cessing is performed in hardware by graphic board, management of interactions between plant and the environment must be done separately. Modeling of tire and road interaction can be mathematically complex as whole vehicle and suspension modeling with the added difficulty to maintain temporal and spatial synchronization between what is simulated and what is displayed by the visual system. Acquisition of pilot commands and creation of a virtual cockpit has smaller requirements in terms of computational load but still needs a strong temporal synchronization with simulator and visual system to reduce action/effect latency.

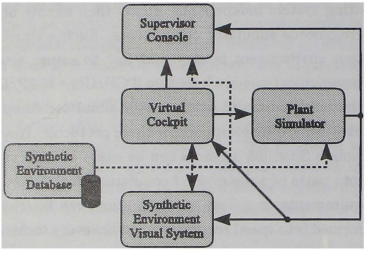


Figure 1: Trainer logic components

PLANT SIMULATOR

Creation of a trainer prototype requires, first of all, a simulator of plant dynamics. Simulator inputs are operator and supervisor commands and external stimuli and disturbances. These signals have different sources and maybe sampling rate; Simulator outputs must be used to create the synthetic environments: plant internal state like position, orientation, velocity is used to correctly place the plant virtual representation in the virtual world; plant acceleration and orientation can be used as input to an eventual moving platform controller and different operating conditions (speed, thrust, tire and road skidding) can be used as input to a digital sound synthesizer to create acoustic feedback. All this components must run time synchronized for a reliable trainer.

Plant simulator must also take into account plant interaction with virtual world; it is hence necessary a feedback loop between plant simulator and virtual world simulator. Synthetic environment visualization must have a mathematical counterpart to be used to provide environmental feedback to the plant simulator. To obtain a full spatial coherence between what is simulated and what is portrayed by the visual system, a common shared database must be used for the synthetic environment creator and the plant simulator.

A detailed plant dynamics simulator can be built with commercial tools like Matlab and Simulink; automatic C code generation is also possible but there is no support at all for manned simulations. Building complex models is rather easy but real time synchronization, input commands acquiring and 3D graphics visualization is very primitive. Furthermore such tools are not optimized for graphic user interface speed during simulations, hence they are not suited for building immersive simulation environments. It is so reasonable to think input acquiring and virtual cockpit generation, dynamics simulator and virtual world creator as three separate tasks. Even supposing an ideal multitasking, that is no context switch delay and no shared resources (video and hardware ports) conflict, running these tasks on one single machine appears irrational. First of all CPU time is divided between the tasks resulting in a big performance drop; then, necessary communications between tasks can have frequency requirements near the minimum lapse of time between context switches and this can be source of a further performance decrease.

INPUT CONSOLES

The operator console is composed of a set of input devices and a virtual cockpit. Input devices like sticks, buttons, knobs are digitally sampled and their state is sent to the plant simulator. A virtual cockpit can be built using Cathode Ray Tubes (CRT) driven by a Personal Computer or by a dedicated hardware. Use of hardware in the loop components can be very useful to increase simulator re-

alism; virtual cockpits are also a fast way to prototype real instrumentation.

A virtual cockpit creation software should provide a set of common instruments, the power to use custom designed instruments coded with a high level language, C++ for example, or a higher level functional/graphical description meta-language. A visual instrument design tool should allow assembling of simple graphic objects: quads, triangles, buttons, pointers, providing animation information to build a more complex instrument. It should finally allow complex panel layout.

Virtual instruments and input status is continuously updated by plant simulator; communication to virtual cockpit and from input console occupies a band proportional to sampling rate and amount of data; this bandwidth can grow very much if a high number of instruments or a low sampling period is used.

Supervisor console gathers data from all other components and allows data storage. Interaction with all components of the trainer permits on the fly change of plant dynamic characteristics, simulate faults, complete control on all simulation parameters. Communication to and from supervisor console does not have strict synchronization requirements because supervisor is not part of the simulation loop and acts asynchronously; in fact, action effect delay should be minimized anyhow.

SYNTHETIC ENVIRONMENT

Creation of a synthetic environment (SE) where the pilot and his plant may move is very complex. A realistic SE must provide sensory information that imitates at best real world sensations, that is, hearing, sight and touch must be deceived to give the operator reality feeling. Only with a realistic SE, such a simulator can be used as a reliable trainer. The main components of a good SE are: photorealistic images, high frame rate and real time.

The need for a photorealistic vision of the world is perhaps the weakest of the three; what is really important is that SE objects have to be easy recognizable and such as not to divert the operator from his/her task. In a demanding flight simulation the pilot may not particularly care about the realism of trees on the land, but will find disturbing an incorrect perspective or any other poorly positioned or dimensioned objects that can distort his perception of distance and depth.

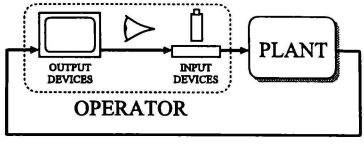


Figure 2: Operator and plant closed loop

An high frame per second value: f_{ps} is vital for a realistic SE; f_{ps} can be considered as the bandwidth of the connection between the plant and the operator and acts as a filter on plant outputs and such a filter in the feedback loop might also lead to instability. This problem may be overcome only by raising "vision filter" cutoff frequency higher or equal to the minimum between operator and plant bandwidth. Moreover the display system introduces a time delay τ_v between action (input signal) and plant reaction (output signal) that is added to the real plant delay such as $max(\tau_v) \leq \frac{1}{f_{ps}}$. It is easy to see that low values for f_{ps} lead to high maximum delays causing operator confusion and maybe instability.

It is not trivial, however, to see that, in fact, there is a certain difference between the frame rate and synthetic objects status update rate: O_{ur}. Obviously $O_{ur} \leq f_{ps}$ because an object cannot be drawn faster than the frame rate, but some data could have a communication rate less than f_{ps} resulting in a narrower update band not for the whole display but only for the object to whom data refers (In fact $O_{ur} = k \cdot f_{ps}, k \in \mathbb{N}^+$). If this object has to be controlled by the operator, the band shrink and associate delay reflects itself on closed loop transfer function and hence on stability.

Finally Real time is necessary because causality must be guaranteed and data loss or queuing must

be avoided. Moreover the presence in the loop of hardware components, that cannot run other than real time, reinforces the need for a real time simulation environment. Hence, for almost realistic 3D rendering of the world very fast graphic processing is needed.

A large and detailed synthetic environment requires a huge amount of memory to store 3D objects geometries and manage all interactions between the plant, sensors, actuators and the SE; a database, shared amongst trainer components, can be used to obtain spatial coherence. Road and wheel simulation is a classic example: simulation of a vehicle running at $80km/h$ with a road surface definition of $10cm$ needs a sampling time smaller than $5ms$, that is, more than 200 integration steps per second; for the simulation of the complex non linear model of the tire, this might be too much. On the other side, 3D rendering of such a detailed road needs very fast graphic hardware and implies a large amount of memory access even though dynamic level of detail techniques can be used to effectively speed up rendering but require additional computations.

When geometric information is used not only for visualization but also for simulation, like in collision detection, an additional data flow between the SE shared database and the simulator must be established. Bandwidth required by this task might be too much: an alternative solution is duplicating the database, or maybe only its task relevant features, on the various machine accessing it. The visual system can draw the scene without any knowledge of actual collisions that, are detected by the plant simulator, or, if high resolution is needed, by a dedicated machine. Even a small delay between collision and contact forces generation causes a virtual body penetration that affects contact simulation precision; hence collision status update rate should be higher as possible.

DISTRIBUTED SIMULATION

In a fine grane simulation environment, trainer components can be distributed over a Local Area Network (LAN); an appropriate deadlock free communication protocol must be developed to allow Real-Time synchronization between simulation entities. This to obtain a low cost hardware solutions by distributing workload amongst various workstation to reach high computational power.

Optimal simulator decomposition can be based on: logic component subdivision by their functions, single component computational needs and bandwidth requirements on connecting channels.

FUNCTIONAL SUBDIVISION

Due to logical trainer components connections not every possible subdivision appears reasonable; Often network subdivision follows closely block scheme used to describe the system on paper. Moving away from this distribution causes an increased number of connections between blocks and thickening of global network. In general, a bigger number of logical connections, increases overall bandwidth requirements, even with same amount of data exchanged, and the risk of conflicts on physical network access; this phenomenon is not present in the abstract case of independent communications but, when all signals are passed on the same physical medium, unpredictable delays arise.

COMPUTATIONAL NEEDS

If network distribution is performed to gain overall computational power, a raw estimate of computation time of a single simulation step is enough to suggest an appropriate subdivision. This choice could reveal a double-edged weapon: first of all network distribution of a pipeline of tasks necessarily breaks logic synchronization introducing a delay in every communication channel that alters propagation time in the pipeline; only if distributed tasks have strong logical parallelism, network distribution can be used with ease. Furthermore a load balancing only distribution could generate very strong synchronization needs and high bandwidth communication channels that are in contrast with next point.

BANDWIDTH REQUIREMENTS

In order to reduce risk of sharing conflicts on physical network, communication channels number should be kept as low as possible. At the same time, to reduce risk of physical network bandwidth saturation, rate of transmission of data packets should be as small as possible. Bandwidth occupation is the product of rate of transmission and data packet dimension; in fact, the real performance affecting parameter is rate of transmission: effective physical network bandwidth occupation is much bigger for small packets and fast repeating transmission than for low rate big packets. Laboratory tests have shown practical independence of peak performance from packet dimension, at least for the values of practical meaning in this application field. Figure 3 shows amount of time spent during communications (in *ms*) for a two point TCP/IP communication test setup; near the values of sample time, that is the reciprocal of communication rate, where the graph reaches the 10000 *ms* threshold, real time performance is no more guaranteed, even without any computation time. UDP/IP shows much better performance as in Figure 4; in fact the use of the two is not interchangeable. Finally Figure 5 show how little influence has data packet dimension on TCP/IP communication and that UDP/IP performances decrease almost linearly with vector width.

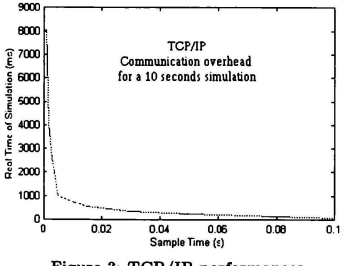


Figure 3: TCP/IP performances

TCP/IP protocol guarantees, via implicit acknowledge, receipt of data packets but occupies a wider network bandwidth than UDP/IP that is much faster but does not guarantee absence of packet losses. Both protocols have been used to implement distributed simulation but a different range of problems have arisen.

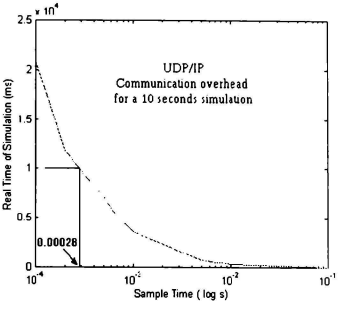


Figure 4: UDP/IP performances

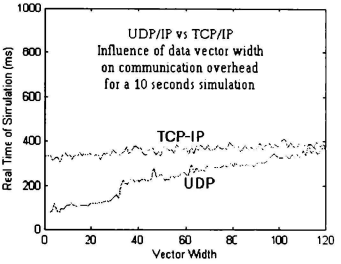


Figure 5: Packet size influence

TCP/IP

Simulation initialization is critical: TCP/IP protocol establishes a virtual connection between two computers in the internet via blocking network access primitives; this task, for a generic computer network where logical loops exists, is not trivial because deadlock situations may arise. A deadlock free communication protocol has been developed

based on a simulation server that allows free initialization sequence of all trainer components; furthermore, recovering of simulation time lost during initialization phase and of temporal coherence between components has been achieved forcing numerical integration algorithms to run with micro-step during this early phase.

UDP/IP

UDP/IP is faster than TCP/IP but does not guarantee receipt of data; this implies the need of an error recovery procedure. A single packet loss in a simulation loop causes a deadlock, this could be avoided applying a timeout in reception but implies real time deadline breaking. To detect packet loss a temporal marker can be added to every packet and appropriate interpolation can be performed to reconstruct lost sample.

If we define a per cent communication overhead as: $\emptyset_c = \frac{\tau_c}{\overline{\tau_e}+\tau_c} \cdot 100$ where τ_c is the time needed to send one sample over the net and $\overline{\tau_e}$ is the maximum value of execution time of one step of the integration algorithm. It is obvious that achieving real time implies: $\overline{\tau_e} + \tau_c < T_s$ where T_s is the sampling period of the connected system, that is the reciprocal of the number of samples sent in one second. It's easy to see that the percentage of time lost during communication increases as T_s decreases, that is when a higher band approximation of system output or input is taken: $\emptyset_c = \frac{\tau_c}{\overline{\tau_e}+\tau_c} \cdot 100 > \frac{\tau_c}{T_s} \cdot 100$.

REAL TIME SYNCHRONIZATION

Several real time policies have been studied. Main issues are synchronization between simulator components and with real time. The latter kind of synchronization is carried on via real time clocks (RTCs); it can be localized on every components with a local RTC or on a subset of them. It can also be performed with a shared RTC only. Synchronization between simulator components can be explicit or implicit. Explicit synchronization happens when there is a communication handshake. The implicit one may happen when communications are between components of a simulation loop or when the communicating elements are all synchronized with their RTCs or with a number of synchronous RTC.

DECENTRALIZED RTC

These synchronization scheme has a RTC on every trainer components. Since every unit has a single RTC, this can be used to explicitly synchronized itself with real time and setup a packet deadline at the end of every communication period. Arises then the need for a recovery procedure when a deadline is broken down; next received packet time marker allows to recognize if error is due to a packet loss or a delay in the sender. In the latter case communication channel buffer must be emptied to allow reading of most recent sample. In the former case, interpolation can be performed between actual and preceding sample. If numerical integration step is smaller than communication step, then simulation only steps can be used to check for a late data packet and eventually apply recovery procedures without loosing any temporal synchronization.

OPTIMIZED RTC

The Decentralized RTCs scheme is, in a certain way, redundant. All components being part of a loop need a RTC only because they are implicitly synchronized by the loop connections: one element of the loop needs preceding element output to perform elaboration and generate subsequent element input. A packet loss produces a loop element blocking and because of implicit synchronization, a whole loop blocking; there is still the need to adopt, like in previous case, a temporal deadline, for packet loss detection. All the elements in an open loop path can be treated as in the Decentralized RTCs scheme.

CENTRALIZED RTC

In this scheme their is a RTC only that sends clock ticks over the network as a broadcast packet to

every simulation components. A data packet is declared lost if not delivered between two clock ticks. The same recovery procedures can be set up. This scheme has the defect of occupying a part of the physical network bandwidth and is more difficult to implement. The rigid inner simulation engine of Simulink has posed serious problems in realization of this scheme.

In fact, all these communication schemes can be mixed together to best fit trainer needs. If TCP/IP is used, then there is theoretically no risk of packet loss and a simpler implementation of communication primitives is possible. If UDP/IP is used, although laboratory tests have shown a pratical zero percentage of packet losses on a dedicated network, the risk of complete simulator blocking in the case of a packet loss inside a loop forces the adoption of temporal deadlines on communications. Anyhow, if all simulator components could perform calculations largely inside the communication period, then the only need would be explicit synchronization with a local RTC to obtain real time performance and explicit synchronization.

The best trainer decomposition in terms of load balancing and network saturation, can be obtained mixing the three methods highlighted. For example, even if distributing a plant and his controller in separate machines may appear reasonable for the first criterion, the plant-controller loop may require be very fast communication (it is enough to think about bandwidth requirements of a relay-based controller) so these two components should be simulated on the same machine; Communications between plant and the 3D visualization subsystem may be easily carried by a local aerea network since they have low frequencies, typically not more than 60Hz).

DYNAWORLDS

DynaWORLDS[1] is an attempt to build a low cost comprehensive distributed simulation and SE visualization toolset carried out at the Department of Electrical Systems and Automation at the University of Pisa.

The distributed simulation tool is composed of a C language library and a Matlab © Toolbox that allows the creation of a network of heterogeneous simulated systems: (C programs or by Matlab). Moreover Matworks Real Time Workshop can be used effectively to generate automatically C code to be used for simulation. Network connections are based on TCP/IP and UDP/IP protocols but the same data stream could be sent on any transmission channel only coding appropriate device drivers. A deadlock free simulation activation protocol has been studied and implemented to allow safe operations.

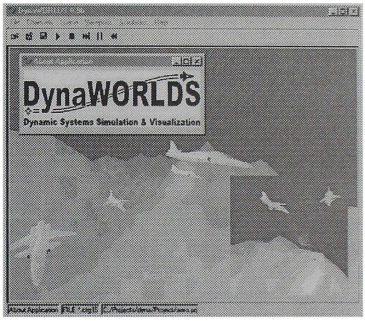


Figure 6: Video Snapshoot

Synthetic environments can be created using an integrated framework of scene design, object animation and control panel design. The world, or scene, can be designed by means of 3D objects, whose geometry and surface proprieties are imported by commercial CAD file formats, lights and cameras. Each object can be connected to a motion channel that affects its position, orientation and scaling in 3D space; can be linked to one other so as to inherit some of its features (a robotic arm) and its position can be tracked with a trail. A link can be established even amongst objects, cameras and

light so that one object can bring cameras (inside vision from a vehicle) and lights (car's headlight). Motion channels are the animation sources for the scene; a channel is the abstraction of a data stream that may have several sources: files, network sockets, I/O boards, input devices like joysticks or buttons. With motion channels it is possible to mix all these sources to obtain very complex object animation. There are also command channels that are used to gather and send operators commands to specified consignees: files, network sockets, etc.

A control panel can then be designed interactively on screen using output devices: camera views, various instruments like pointers, light indicators, artificial horizons etc. .

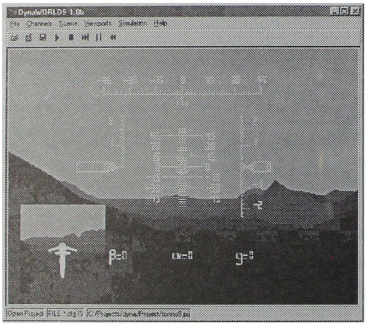


Figure 7: **An aircraft simulator with HUD**

DynaWORLDS is capable of drawing non fixed geometry objects too; trails, smokes, clouds or augmented reality typical tools like guidance tube or data superimposed to recognized objects on the screen can be draw using appropriate graphical plug-ins. Furthermore, particular transformation like nonlinear scaling, squeezing (useful for displaying collisions between elastic objects), bending (vital for representation of flexible structures) are only possible with a custom software.

In the end a complete control over the final rendering makes environmental features like viewing trough fog or turbid water or even the reproduction of night vision devices images feasible.

A DynaWORLDS working demo will be available at the time of the conference.

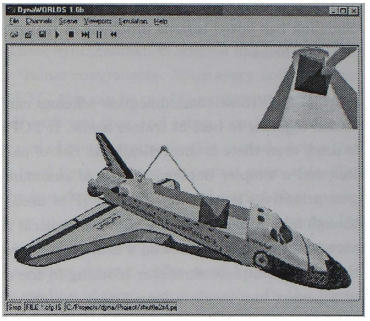


Figure 8: **Space Shuttle robotic arm trainer**

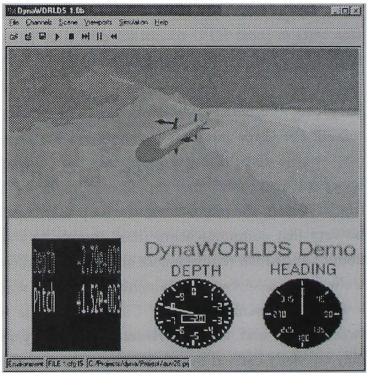


Figure 9: **AUV mission control station**

OPTIMIZING THE ANIMATION LOOP

A critical implementation point of the real time visualization system is the animation loop. Figure 10 shows the animation loop adopted by DynaWORLDS. This algorithm has been designed to

minimize visual information latency respect to real world events; more precisely, in the case of data underrun, the old values are maintained, and in the opposite case, the new frame is not displayed until the exact time is reached. In the ideal case the only delay between an external event and its effect is $1/f_{ps} + T_{SwapBuffers}$ that is the minimal theoretical delay. In fact, this value is corrupted by an additional delay due to an hardware problem: the need of synchronization between the real time clock that drives the simulation and the vertical retrace of the CRT (cathode ray tube); the maximum for this delay value is $1/f_{VerticalRetrace}$.

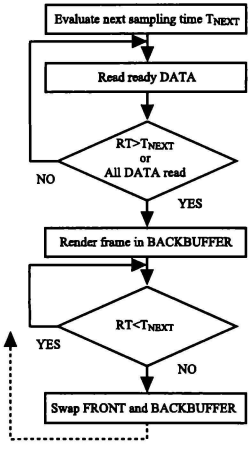


Figure 10: Animation Loop

CASE STUDY

Figure 11 presents a case study: a real time simulator of an Underwater Vehicle (UV), with 2 cooperating robotic arms. The UV is operated by a 3D joystick and the robotic arms by two 6 DOF input devices(two DLR SpaceMouse). The vehicle dynamics simulator, the thruster simulator, command console, arms simulators, simple grasping and collision manager and 3D visualization tool are elaborated on various machines connected by a LAN.

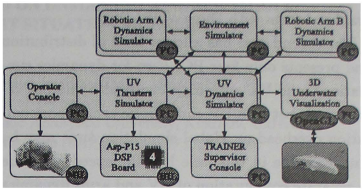


Figure 11: UV Trainer scheme

Real-Time operation is achieved by a proper task distribution amongst computers and high realism, high frame rate world visualization is performed on a dedicated workstation.

Figure 11 shows a block diagram of the simulator's assembly with two different level of computation distribution.

The fine grane simulator is made of eight PCs (six for dynamics simulation): the *"Operator console"* PC is connected to the three input devices and performs command acquisition and realizes a virtual cockpit with DynaWORLDS instrument panel design capabilities. The *"Thruster Simulator"* PC carries a quad SHARC DSP board by Transtech: the PC simulates UV thruster system dynamics and the DSP board runs the nonlinear controller code in parallel. The block *"UV Dynamics Simulator"* and the two *"Robotic Arm X Dynamics Simulator"* have been designed using a robotic cad: Robotics Developer Studio (RDS)[2] and are simulated by Simulink where hydrodynamics effects have been added. Communication rate between these blocks is very high because of rigid connection between arms and UV simulation. The *"Environment Simulator PC"* is used to manage UV and arms interactions with underwater world that is collision detection, sensors simulation and current flow direction database. Because of low speed of vehicle, channels from this block do not have very high bandwidth. The *"3D Underwater Visualization"* PC carries an OpenGL accelerated board and performs all underwater world visualization with DynaWORLDS. The *"Supervisor Console"* gathers all

simulation data and displays them in numerical and graphical form with DynaWORLDS.

The second case has a lower level of distribution and presents only four PCs (two for dynamics simulation), and the only high bandwidth channel remaining is between UV and robotic arms; communication overhead on global simulation time may be not negligible in this case but the speed gain due to separate computation of UV and arms dynamics has overcome network related problems. Simulation tests have shown a fast and easy reconfigurability due to its simple Simulink and C language based structure and Real Time simulation has been performed on an ethernet network of various PC compatible in both configurations.

CONCLUSIONS

This research aims at showing that it is possible to build an affordable and, at the same time, reliable trainer prototype for a plant given all its dynamics simulators, the appropriate input devices, a 3D synthetic environment creator and a safe and fast intercommunication network that provides Real Time synchronization and data exchange between simulator modules. First laboratory tests have already shown feasibility of this work.

*

References

1 Innocenti M. Pollini L. "A Synthetic Environment for Simulation and Visualization of Dynamic Systems" *1999 American Control Conference, San Diego, California, June 1999*

2 G. Casalino, D.Angeletti, G. Cannata, G. Marani "Symbolic-based Model Builder for Multibody Systems" *Fifth International Conference on Information Systems Analysis and Synthesis (ISAS'99), Orlando, USA, August, 1999*

3 Stasko J.C. Crandell R.L. Dunn M.T. Sureshbabu N. Weber W. "Versatility and the hardware-in-the-loop paradigm: moving away from the ad hoc laboratory" *1998 American Control Conference, Philadelphia, Pennsylvania, June 1998*

4 Woon-Sung Lee Jung-Ha Kim Jun-Hee Cho "A Driving Simulator as a Virtual Reality Tool" *IEEEE International Conference on Robotics & Automation 98, Leuven, Belgium, May 1998* pp. 71-76

5 D. P. Brutzman "Virtual World Visualization for an autonomous underwater vehicle" *IEEE Oceanic Engineering Society conference OCEANS 95, San Diego California, October 10-12, 1995*

6 D. P. Brutzman "From virtual world to reality: designing an autonomous underwater robot" *AAAI Fall Symposium on Applications of Artificial Intelligence to Real-World Autonomous Mobile Robots, Cambridge Massachuttes, October 23-25, 1992* pp. 18-22

7 D. P. Brutzman Y. Kanayama M. J. Zyda "Integrated Simulation for Rapid Development of Autonomous Underwarter Vehicles" *IEEE Oceanic Engineering Society Autonomous Underwater Vehicles (AUV) 92 conference, Washington DC, June 4-5, 1992* pp. 3-10

HYBRID FLIGHT SIMULATOR BASED ON UNMANNED AERIAL VEHICLE AND AUTONOMOUS FLIGHT SITUATION MODELING

Sergiy Sadovnychiy, Ph.D., Mexican Petroleum Institute (IMP), Mexico D.F., Mexico.
Ivan Burdun, Ph.D., Georgia Institut of Tecnology, Atlanta, GA, USA.
Vladimir Krivtsov, Dr. of Techn. Science, State Aerospace University, Kharkov, Ukraine
Tomas Ramires, Mexican Petroleum Institute (IMP), Mexico D.F., Mexico.

ABSTRACT

A hybrid technology of flight simulation for pilot training is proposed. The aircraft flight dynamics is represented by a flying model instead of a mathematical model implemented in a ground based simulator. The pilot controls the flying model remotely, from a ground station. The model is an unmanned aerial vehicle (UAV) which is dynamically similar to the modeled aircraft. It has video cameras to record out-of-the-cockpit views which are transmitted to and displayed in the ground station in real time. An autonomous flight situation model is employed as a flight scenario planning tool and a backup flight control method.

Two techniques are described: a method for securing the dynamic similarity between the base UAV and the modeled aircraft and a method for implementing the autonomous flight situation model. Several variants of hybrid flight simulator are proposed. Technical problems of concept implementation and possible solution approaches are discussed. The overall objective is to make flight simulators more affordable and increase the quality of training and the fidelity of flight modeling.

GROUND BASED AND FLYING SIMULATORS

In this section advantages and shortcomings of ground based and in-flight flight simulation methods are briefly analyzed.

Flight simulators and pilot training. The ideal method for pilot training is actual flight on the aircraft type which will be flown in operation. However, due to cost, safety and other constraints, flight simulators have to be used as a substitute.

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

The role of training flight simulators is to develop basic airmanship skills and practical flight related knowledge in students, as well as to maintain and upgrade advanced flight control skills in professional pilots. (Sometimes the term 'flight management' is used instead of the 'flight control').

Training flight simulators can be divided into two classes: ground based and flying.

Ground based simulators. The simplest flight simulation technology is a software program for a personal computer equipped with a device to imitate flight control inputs. There is a variety of professional flight simulators[1] ranging from functional imitators of specific onboard systems to sophisticated full scale modeling complexes. The most advanced are flight simulators used by airlines and military and reconfigurable engineering research simulators.

Shortcomings. The ground based flight simulation technology is safe and less expensive training means compared with actual aircraft. However, it has a limited flight modeling capability. The shortcomings include the following: (1) limitations on out-of-the-cockpit view modeling, (2) a limited capability of flight scenario planning and execution, and (3) a limited fidelity of flight dynamics modeling.

Flight visualization techniques. To address the first problem the following visualization techniques are employed. First, a mockup of some landscape is built in miniature. During simulated flight, this terrain model is scanned by a tele-/videocamera, and the image is projected on a screen in front of the simulator's cockpit. Second, a pseudo-realistic image of out-of-the-cockpit views can be generated on a computer using various graphic primitives, such as fields, trees, buildings, runways, towers, etc. Finally, a digital terrain map of a limited geographic region can be constructed based on the

GPS data. Then, using advanced mathematical methods (e.g.: fractals) and computer graphics, realistic surface details are added. However, these techniques require a substantial amount of preparatory work or/and computer processing power and are very expensive.

Flight scenario planning techniques. In a flight simulator training scenarios are planned and controlled by a pilot instructor or/and using special software and hardware. (Note: we will consider two kinds of flight scenario - demonstrative and training.) A demonstrative scenario is basically an example of some flight situation-case (e.g.: correct/incorrect piloting tactics, incident or accident). Normally, these situations are pre-recorded and then demonstrated to the student. A training scenario is a plan of an interactive flight situation, which may include non-standard flight events and demanding operational conditions. Unlike demonstrative scenarios, student's responses to these circumstances constitute a part of training scenarios. One of the shortcomings of the scenario planning techniques used in flight simulation is a limited "what-if" experimentation capability and a lack of human pilot decision making models.

Flight dynamics modeling. At present, the equations of aircraft motion are available in the most generic form [10]. There are also efficient numeric techniques [11] to solve these equations. Models of the onboard systems which may affect the aircraft flight dynamics and flight control (sensors, actuators, avionics, etc.) are normally known, e.g.: from previous prototypes, etc. Empirical-theoretical models of external operational conditions, such as rain, ice, runway condition, wind, etc., are available as well [12,13].

However, the main method for achieving a higher fidelity of flight dynamics modeling is to improve the quality of the aircraft input characteristics. This is not always possible as aircraft manufacturers may not want to release all the characteristics of their products, especially the data related to extreme flight regimes.

In-flight simulators. There are several kinds of in-flight (flying) simulators. This list includes trainers, specially equipped operational aircraft, and flying laboratories. A trainer is the most popular kind of flying simulator. The role of trainers is to develop in students basic piloting skills and practical knowledge of flight dynamics and control.

The highest fidelity of in-flight flight dynamics modeling can be achieved when the base vehicle is of the same type as the operational aircraft or in a special flying laboratory. The principle of operation of all the flying simulators is to modify apparent stability and control responses of a base vehicle by feeding back response parameters into its electrical flight control system, and by shaping the pilot's inputs[2]. One of the advantages of in-flight simulation is that the pilot acts in his (her) natural operational environment with actual visual cues and actual motion stimuli of the modeled aircraft. Another advantage is that there is no need for a comprehensive mathematical model of the aircraft flight dynamics. Normally, the base vehicle has better dynamic characteristics than the modeled one. So it is not a technical problem to deteriorate its performance in order to match the behavior of the modeled aircraft.

However, due to cost and safety constraints a high fidelity flying simulators have very limited applications. They are used mainly to finalize a training course, assess the flight performance of new aircraft, and for other unique purposes.

Therefore, both ground based flight simulators and flying (in-flight) simulators used for pilot training have advantages and shortcomings. The main problem is how to achieve a higher fidelity of flight modeling and improve the quality of pilot training with minimum expenses.

RESEARCH TASK FORMULATION

Problem. The problem under study can be formulated as follows. Given a task of flight simulation for pilot training, how to (1) increase the fidelity of modeling of the aircraft flight dynamics, (2) achieve a better quality of visualization of out-of-the-cockpit views, and (3) enhance the demonstrative and training scenario planning capability?

The overall objective is to reduce the cost of flight simulation and pilot training with a simultaneous increase in the quality of training.

The solution approach. A new method for addressing this problem is proposed. It is suggested to employ a hybrid flight simulator as a substitute for the "pilot – vehicle – operational conditions" system. The simulator includes a flying model (UAV), the pilot, a ground station, a situational flight model, a synthetic environment of actually present and simulated operational conditions, and auxiliary software and hardware.

Main principle. The central element of a hybrid simulator is an unmanned aerial vehicle, which is radio controlled by the pilot from a ground station. The flight control system of the UAV imitates the flight dynamic characteristics of a specific aircraft. Onboard digital video cameras provide a real-time picture of out-of-the-cockpit views of flight. This information, together with the current flight state parameters, is transmitted to the ground station via a radio channel. A virtual reality Head-Mounted Display[3] is used to present a realistic out-of-the-cockpit panorama of flight to the pilot. This picture is combined with a computer generated image of the instrument panel of the modeled aircraft.

Also, an autonomous flight situation model[8,9] is used on board the flying model and in the ground station. Its purpose is to provide a backup flight control in emergencies or under pilot's request and automate the process of planning of demonstrative and training flight scenarios. Depending on the scenario, the model can perform either autonomous or joint (i.e., together with the pilot) flight control.

Potential advantages. One of the potential advantages of hybrid flight simulators is that the pilot has almost unlimited freedom in executing flight maneuvers in any direction and over any terrain. The presence of the autonomous flight situation model in the control loop helps make training more flexible and safer. Also, external cues which the pilot receives from the flying simulator are almost identical to actual flight (perhaps, with the exception of the takeoff and landing modes – see the discussion section below).

Finally, the cost of the airframe and engine installation of a modern unmanned aerial vehicle (excluding onboard military or other special equipment or payload) is much lower compared with the price of a high-fidelity ground based flight simulator. Other expenses - for the ground station, HUD, onboard video cameras, etc. - are also relatively low.

SIMILARITY OF MOTION

The problem of securing the similarity of motion between the UAV and the modeled aircraft is a problem of finding a control law for the base UAV as a function of its motion parameters[4].

Equations of motion. Two systems of the equations of motion are considered: one is for the base UAV and another - for the modeled aircraft (the index m stands for the modeled aircraft):

$$\dot{x} = F(x, u, \varphi) \tag{1}$$

$$\dot{x}_m = F_m(x_m, u_m, \varphi_m) \tag{2}$$

where x is a state vector, u is a control vector, φ is a vector of external disturbances, and F is a vector-function.

It is assumed that the dimensions of the appropriate vectors in (1) and (2) are the same, and all the components of x can be measured without error.

Similarity conditions. The conditions of similarity of the flight dynamics properties for these two vehicles can be formulated as follows. Given the same initial conditions, i.e.:

$$x(t_0) = x_m(t) \tag{3}$$

- there exists a control $u(t)$, $u(t) \in G$, which provides for $\forall\ t \geq t_0$ the equality of the state vectors for the two vehicles, namely:

$$x(t) = x_m(t) \tag{4}$$

for $(\forall u(t_m))\ (u_m(t) \in G_m;\ \sqcup(t) \in \Phi,\ \sqcup_m(t) \in \Phi_m)$, where G and G_m - are the domains of possible control inputs for the flying model and the modeled aircraft, respectively; Φ and Φ_m - are the domains of possible external disturbances. Therefore, the problem can be reduced to a study of the implementability of the motion $x(t)$, given the goal program $x_m(t)$. Note that the program $x_m(t)$ is a solution of the system (2), which describes flight dynamics of a specific (modeled) aircraft.

Thus, the task of securing the similarity of motion between the UAV and the modeled aircraft is to find a control law $u(t)$, which implements the relationship (4). It has been demonstrated[4] that if the UAV motion can be described by the system:

$$\dot{x} = Ax + Bu \tag{5}$$

and the motion of the modeled aircraft - by the system

$$\dot{x}_m = A_m x_m + B_m u_m \tag{6}$$

and if there exists a control law $u(t)$, then the following statement will be true:

$$\dot{x}(t) = \dot{x}_m(t). \quad (7)$$

Thus,

$$Bu = A_m x_m + B_m u_m - Ax. \quad (8)$$

After adding $(Ax_m - Ax_m)$ to the right part of (8) and taking into account the condition (4), we get the following:

$$Bu = (A_m - A)x_m + B_m u_m. \quad (9)$$

Similarly, by adding $(A_m x - A_m x)$, we get:

$$Bu = (A_m - A)x + B_m u_m. \quad (10)$$

The equations (8), (9) and (10) give the following three variants of the control law to satisfy the condition (4):

$$u = B^+ A_m x_m + B^+ B_m u_m - B^+ Ax; \quad (11)$$

$$u = B^+ (A_m - A)x_m + B^+ B_m u_m; \quad (12)$$

$$u = B^+ (A_m - A)x + B^+ B_m u_m. \quad (13)$$

where $B^+ = (B^T B)^{-1} B^T$ is a pseudo-inverse matrix and the index T means transposing.

The choice of control laws from the list (11)-(13) depends on the application task. The first law provides the invariance of the UAV to wind disturbances by converting the equations of its motion to the equations of a neutral body. The law (12) difffers from the first one in the use of signals from the model which implies that measurements of flight parameters for the UAV are not required during flight. In the third law (13) there is no need to solve the equations of motion (6) for the modelled aircraft – it is sufficient to measure flight state parameters for the UAV.

Similarity by airspeed. One of the problems of securing the similarity of motion between the UAV and the modeled aircraft is the similarity by speed. Many UAVs (such as "Pioneer", "Hermes", etc.) have a maximum cruise speed of up to 300 km/h. This covers the majority of flight regimes for general aviation aircraft, as well as the takeoff and landing phases for other airplane types. To model flight of large (subsonic) aircraft, special UAVs are required to be able to address other flight regimes, such as climb, descent and cruise. Nevertheless, the production cost of such UAVs is expected to be much lower than for military UAVs.

One of the most complex tasks is to adequately model flight of highly manueverable airplanes, such as F-18, Sukhoj-27, Rafale, and F-22. The use of special electronic modules adjusting the dynamic characteristics of the vehicle control system does not help meet the similarity criteria for such aircraft types at high angles of attack, during rapid turns, etc. In these cases, it is required in addition to secure the aerodynamic similarity between the UAV and the modeled aircraft. For this purpose a large scale dynamically similar [flying] model, or DSM, can be used as a base vehicle. At present, DSMs are broadly used for advanced flight control concept research [5,6]. For example, DSM X-36 has a full set of equipment to remotely control the vehicle by the pilot from the ground.

A DSM has the same aerodynamic configuration as the modeled aircraft. Normally, the structure of its control system is similar to the original aircraft system. In all cases, the coefficients required to obtain model's gains from the aircraft's ones depend on actual dimensions and are defined by the following equations[7]:

$$K = 1, \text{ for dimension } \left[\frac{\text{degree}}{\text{degree}}\right]$$

$$K = \sqrt{K_L}, \text{ for dimension } \left[\frac{\text{degree}}{\text{degree / s}}\right]$$

$$K = 1/\sqrt{K_L}, \text{ for dimension } \left[\frac{\text{degree / s}}{\text{degree}}\right]$$

where $K_L = \frac{L_M}{L}$; K_L is the linear dimension scale.

To achieve the required stability and controllability characteristics in a DSM-based flying simulator the Froude similarity criterion must be met. The equation expressing the required relationship between inertial and weight forces is as follows:

$$\frac{V_N^2}{gL_N} = \frac{V_M^2}{gL_M} = Fr; \text{ and } Fr = idem \quad (14)$$

where V is the velocity, g is the acceleration due to gravity, and L is the characteristic length. Finally, the following conditions of compliance with the Froude criterion for the velocity and time complete the system:

$$V_M = V \cdot K_V = V \cdot \sqrt{K_L}$$

$$t_M = t \cdot K_t = t \cdot \sqrt{K_L}$$

Note that because K_L is always less than 1, all the processes in the flying model run faster. Thus, if a DSM is used, special modules must be introduced into its flight control system to slow down the dynamics of control surfaces (the rates of deflection, etc.). This issue requires a more detailed study.

Thus, employing a dynamically similar flying model, a higher fidelity of physical modeling of the aircraft aerodynamics and flight dynamics can be achieved. In addition, various failures in the flight control system (e.g.: actuator's malfunctions), as well as mechanical damages to the airframe and control surfaces, can be imitated.

AUTONOMOUS FLIGHT SITUATION MODEL

In this section an introduction is made to the autonomous flight situation model. This model is proposed as a method for flight scenario planning and backup flight control in a hybrid flight simulator. An example of flight scenario will be given to demonstrate how this method can be implemented.

The autonomous flight situation model. Basically, the autonomous flight situation model is a system of data structures and generic computational algorithms which model a flight situation as a discrete-continuous cause-and-effect structure.

The object of autonomous flight modeling is the behavior of the 'pilot - vehicle - operational conditions' system in a complex flight situation. Modeled phases and modes of flight include: take-off (normal, aborted, and continued), landing (normal, continued, and go-around mode), climb, descent and landing approach (any profile), en-route flight modes (any profile), groundroll motion, and special/test maneuvers (e.g.: stall, spin, power-off flight, etc.).

Components. The autonomous flight situation model includes the following components:

- a situational, or tactical, pilot model ('silicon pilot')
- models of selected external factors (e.g.: wind shear, rain, icing)
- models of selected internal factors (e.g.: engine failures, control surface hardovers, pilot errors).

In this paper the first component, a situational pilot model, will be introduced.

Situational pilot model. The situational pilot model is a system of input flight scenarios and computational algorithms that imitates a limited subset of a human pilot's knowledge and decision making functions required to perform situational (tactical) flight control.

The pilot's control tactics are formalized at the level of cause-and-effect relationships between flight events and control processes. These are the only two object types, which are required to plan and simulate flight situations of practically any complexity [9].

Limitations. The proposed pilot model has the following limitations. Pilot's sensor-motoric functions are not modeled. Pilot's strategic decision making functions are not modeled with the exception of the flight scenario planning function. Situational decision making is formalized as a multi-stage control process based on flight scenarios.

Why to model the human pilot? The role of adequate situational models of the human pilot in training is very important. Situational models provide a link between the perceptual-motor and strategic levels of a human pilot's decision making mechanism. Situational (tactical) control is largely responsible for safe and unsafe outcomes of a particular flight. Using such models, pilot's tactics and errors can be analyzed as an integral part of the system behavior. Finally, the situational pilot model is a powerful tool for studying an emerging class of flight safety problems - multi-factor operational domains of flight and "chain reaction" flight accidents [9].

There are three basic concepts of the autonomous flight situation model: the flight event, the flight process, and the flight scenario. These concepts permit a uniform formalization of the majority of phases, modes and conditions of flight for all aircraft types [9].

Flight event. The flight event is a special state of the "pilot - vehicle - operational conditions" system, which is important to the pilot and stands for a substantial change in the current flight situation. Examples of flight events are as follows: *"left engine out", "speed VR achieved", "altitude 360 ft and speed 180 kn", "on the runway", "high angle of attack", "30° left bank", "go-around decision"*, etc.

Main classes of flight events include: independent and dependent (in the latter case an "if-event", or event-condition, is checked first), simple and compound (determined by the number of components in the event recognition criterion), precise and fuzzy (determined by the type of variable in the criterion), momentarily recognizable and recognizable with a delay, unique and periodical (repeating), single and serial. These class pairs may have non-empty intersections.

Calendar of flight events. The list of all the events which may occur in a particular situation, or in a group of situations, is called the flight event calendar, $\Omega(\mathbf{E})$:

$$\Omega(\mathbf{E}) = \Omega^{NR}(\mathbf{E}) \cup \Omega^{JR}(\mathbf{E}) \cup \Omega^{F}(\mathbf{E}) \cup \Omega^{P}(\mathbf{E}), \quad (15)$$

where $\Omega^{NR}(\mathbf{E})$ is a subset of "not recognized" events, $\Omega^{JR}(\mathbf{E})$ - "just recognized" events, $\Omega^{F}(\mathbf{E})$ - "frozen" events, and $\Omega^{P}(\mathbf{E})$ - "past" or "recognized" events. The symbols {NR, JR, F, P} stand for the four possible subset-states of a flight event during simulation. Note that $\Omega^{A}(\mathbf{E}) = \Omega^{JR}(\mathbf{E}) \cup \Omega^{F}(\mathbf{E})$ is a subset of "active" events.

The flight event calendar may be viewed as a discrete logical framework to which various flight processes are attached. Graphically, flight events are depicted as ellipses or circles with the event name and code.

Event recognition criterion. A flight event becomes active in the situational model if its recognition criterion is true:

$$(((x \square R)_1 l_{12} (x \square R)_2 l_{23} (x \square R)_3 \ldots) - \text{true}) \Rightarrow (\mathbf{E} \in \Omega^{A}(\mathbf{E})), \quad (16)$$

where x is a flight variable, $x \in V$, V is a vocabulary of all the flight variables; ij $\in$ {12, 23, ...} - pair number; $l_{ij} \in$ {OR; AND} - logical link between the elementary criteria $(x \square R)_i$ and $(x \square R)_j$; $\square \in$ {GT, LT, EQ, BEL, GE, LE, NE, AE, NA} - relation type[8]; R - right part of the criterion, $R \in$ {**a**; [**a**; **b**]}, **a** and **b** - real numbers, **a**<**b**.

For example, a compound event $\mathbf{E}_4$: *"at circuit altitude"* can be activated using the following recognition criterion: (H = 1200) AND (V_z BEL [-1.0; 1.0]).

Flight event specification. In the situational model flight events are defined using the following generic frame-specification:

$$R[\mathbf{E}_i] = \{ i, j^{IF}, \mathbf{N}, (x_1, \ldots, x_n), (x \square R), \Delta t, \tau \}, \quad (17)$$

where i is the event code, i $\in$ {1, 2, ...}; j^{IF} - code of the "if-event" (event-condition), $j^{IF} \neq i$; N - event name, $(x_1, \ldots, x_n)$ - list of variables to be memorized when the event is recognized, $x_i \in V$, $(x \square R)$ - recognition criterion, Δt - time period (for periodic events), $\Delta t \geq 0$; τ - required delay in the event recognition process (after the recognition criterion becomes true), $\tau \geq 0$.

Example. The following input frame specifies an event E_3 for modeling: R[$\mathbf{E}_3$] = { 3, 1, "speed VR", (3,19,14,1), (77 AE 290.0), 0.0, 0.5 }. It means that the event $\mathbf{E}_3$: *"speed VR"* will be recognized when the indicated airspeed (the variable $x(77)$) will reaches some 290 km/h. Current values of the flight variables {$x(3)$, $x(19)$, $x(14)$, $x(1)$}, or {δ_e, L, θ, α}, will be memorized when the event is recognized. There is also a required 0.5 s delay in recognizing the event.

Flight process. The flight process (Π) is basically a time-history of one or several flight state variables, which characterize a certain aspect of the behavior of the "pilot – vehicle – operational conditions" system. Flight processes are used to formalize dynamic properties of the vehicle, flight control tactics including human piloting and errors, functions and malfunctions of onboard systems, and weather conditions. Every flight process has its specific purpose in the cause-and-effect structure of a flight situation. Unlike the events, the processes are continuous components of the situational model.

The following phrases represent various flight processes: *"keep to runway centerline"*, *"keep pitch at 10° during initial takeoff climb"*, *"windshear 10 ft/s per 30 ft of altitude"*, *"r.p.m. decay after engine #1 failed"*, *"flaps down from 0° to 15°"*, *"turn at 20° bank and zero sideslip"*, *"wet runway"*.

Flight processes are graphically depicted as arrows marked with process main attributes (type, name, code, and other).

Flight process types. Flight processes can be organized by their nature and purpose into the following groups: vehicle dynamics (**D**), flight control processes (**T**, **O**, **P**), airborne systems functioning and failures (**B**, **F**), external operational conditions (**A**, **R**, **W**, **Y**, ...), and some other[8,9].

The united list of flight processes, $\Omega(\Pi)$, can be written as follows:

$$\Omega(\Pi) = \Omega(\mathbf{T}) \cup \Omega(\mathbf{P}) \cup \Omega(\mathbf{O}) \cup \Omega(\mathbf{B}) \cup \Omega(\mathbf{F}) \cup \Omega(\mathbf{W}) \cup \Omega(\mathbf{R}) \cup \Omega(\mathbf{Y}) \ldots \quad (18)$$

where $\Omega(\mathbf{T})$ is a subset of piloting tasks, $\Omega(\mathbf{P})$ - control procedures, $\Omega(\mathbf{O})$ - state "observers", $\Omega(\mathbf{B})$ -onboard system functions, $\Omega(\mathbf{F})$ - onboard system malfunctions, $\Omega(\mathbf{W})$ - wind conditions, $\Omega(\mathbf{R})$ - rain conditions, $\Omega(\mathbf{Y})$ - runway surface conditions, etc.

In the situational pilot model the first four types are used, namely: piloting tasks, flight state observers, control procedures, and onboard system malfunctions. Obviously, a subset of flight control processes includes piloting tasks, state 'observers', and control procedures.

Piloting task. The piloting task (**T**), or the task, is a manual flight control process. It is carried out using primary controls (elevator, ailerons, rudder, etc.). Piloting tasks represent control with feedback. Every piloting task requires observation of the current flight state modeled by state 'observers' (see below). The examples of $\mathbf{T}_j$ are as follows: $\mathbf{T}_4$: *"keep to the centerline during groundroll"*, $\mathbf{T}_5$: *"make coordinated turn at bank +15°"*, $\mathbf{T}_8$: *"keep pitch at 10° and zero bank during initial climb after liftoff"*.

Input specifications of piloting tasks and other control processes are described in [8].

State 'observer'. The flight state 'observer' (**O**) is the process of evaluation of current states of the system and comparison of these states with the relevant tactical objective (goal state). The aim is to detect an error between these two states sufficient to change the performance of the piloting task. For example, the piloting task $\mathbf{T}_8$ listed above is provided with a state 'observer' $\mathbf{O}_1$ to monitor the vehicle motion in pitch. This 'observer' may include the following three components (elementary state 'observers'[8]) used to monitor pitch angle, pitch rate and pitch acceleration: $\mathbf{O}_1$ = (`PitchObs`, `PitchRateObs`, `PitchAccelObs`).

Control procedure. The use of secondary controls (flaps, spoilers, etc.), as well as single movements with the primary controls, are described by the process type called control procedure (**P**). For example, $\mathbf{P}_1$: *"wheels – up"*, $\mathbf{P}_2$: *"unstick"*, $\mathbf{P}_3$: "flap 30°→15°", $\mathbf{P}_6$: *"engines - to MCPR"*.

Failure. The onboard system's failure is a process which imitates abnormal function of some onboard system. The examples are as follows: $\mathbf{F}_2$: *"left engine failure"*, $\mathbf{F}_8$: *"uncommanded deployment of thrust-reverser"*, $\mathbf{F}_{27}$: *"elevator jammed at 17.5°"*. In the situational pilot model, failures are formally described as artificial control procedures and thus may constitute a part of a demonstrative or training scenario.

Flight process states. During simulation every flight process from $\Omega(\Pi)$ can be in one of the following three subset-states: $\Omega^{NO}(\Pi)$, $\Omega^{O}(\Pi)$, or $\Omega^{CL}(\Pi)$, i.e.:

$$\Omega(\Pi) = \Omega^{NO}(\Pi) \cup \Omega^{O}(\Pi) \cup \Omega^{CL}(\Pi), \quad (19)$$

where $\Omega(\Pi)$ is the united list of flight processes, $\Omega^{NO}(\Pi)$ - "not open" processes, $\Omega^{O}(\Pi)$ - "open" processes, and $\Omega^{CL}(\Pi)$ - "closed" processes; $\Omega^{O}(\Pi) = \Omega^{A}(\Pi) \cup \Omega^{F}(\Pi)$, $\Omega^{A}(\Pi)$ - "active" processes and $\Omega^{F}(\Pi)$ - "frozen" processes.

Flight scenario. The flight scenario (**S**) is a plan of a flight situation. It formalizes the content and the logic of flight including flight control. The flight scenario **S** is formed of two sets of objects - flight events, $\Omega(\mathbf{E})$, and flight processes, $\Omega(\Pi)$. They represent, respectively, discrete and continuous components of the flight situation model.

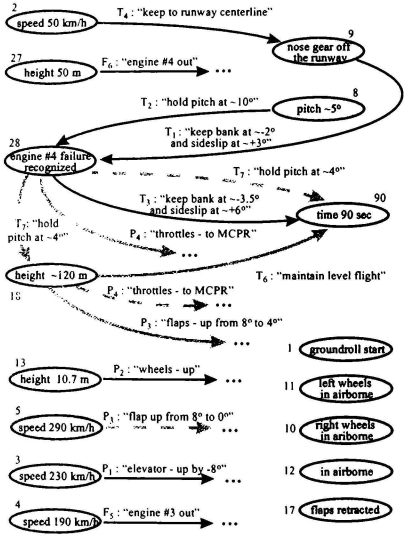
Scenarios may be depicted as directed graphs with the flight events as vertices and the flight processes as arcs. Examples of scenarios are as follows: $\mathbf{S}_1$: *"Normal takeoff"*, $\mathbf{S}_3$: *"Aborted takeoff with left engine out"*, $\mathbf{S}_{12}$: *"Groundroll on wet runway"*, $\mathbf{S}_7$: *"Takeoff with two right hand engines out"*, $\mathbf{S}_{10}$: *"Stall in takeoff configuration"*, $\mathbf{S}_{19}$: *"Cruise mode at 500 kn and 30,000 ft"*.

Example. The event-process flight description language can be used to program both demonstrative and training scenarios. A realistic example of such a scenario is shown in Fig. 1.

This is a complex flight situation $\mathbf{S}_7$: *"Takeoff of a four engine airplane with two right hand engines out"*. Both correct and incorrect control tactics are shown. Following is a description of a correct piloting method under the given conditions.

The situation starts on the runway, at the flight event $\mathbf{E}_1$: *"ground roll start"*, with brakes released and throttles at full power. When the airspeed reaches about 50 km/h (event $\mathbf{E}_2$), a piloting task $\mathbf{T}_4$: *"keep to runway centerline"* is initiated by means of rudder and a nose wheel control system until the nose wheel is on the ground (event $\mathbf{E}_9$).

At the airspeed of ~190 km/h (event $\mathbf{E}_4$) an artificial control procedure-failure $\mathbf{F}_5$: *"engine #4 failed"* is introduced. At a rotation point (event $\mathbf{E}_3$: *"speed VR = 230 km/h"*) the appropriate action is applied by elevator, $\mathbf{P}_1$: *"move elevator up by -8°"*. When the



Legend:
- correct control process
- incorrect control process

Fig. 1. Training flight scenario example $\mathbf{S}_7$: "Takeoff with two right hand engines out"

pitch angle exceeds ~5° ($\mathbf{E}_8$) the 'silicon pilot' initiates a piloting task $\mathbf{T}_2$ to hold pitch attitude at about 10° by elevator. Also, after the nose wheel leaves the ground ($\mathbf{E}_9$), the pilot tries to maintain the command bank and sideslip angles required to help counteract the effect of thrust asymmetry ($\mathbf{T}_1$: *"keep bank at -2° and sideslip at +3°"*) by ailerons and rudder. Wheels are retracted at event $\mathbf{E}_{13}$: *"height 10.7 m"* by means of control procedure $\mathbf{P}_2$: *"wheels - up"*.

When the engine #4 failure is planned to occur (at event $\mathbf{E}_{27}$: *"height 50 m"*) another artificial control procedure is activated, $\mathbf{F}_6$: *"engine #4 failure"*. In two seconds (at event $\mathbf{E}_{28}$: *"engine #4 failure recognized"*) the 'silicon pilot' sets throttles ## 1 and 2 to a maximum continuous power rating (procedure $\mathbf{P}_4$). Simultaneously, the model decreases the command pitch angle to reduce drag (task $\mathbf{T}_7$: *"hold pitch at +4°"*). Also, it further adjusts the command bank and sideslip angles to account for deteriorated thrust asymmetry (task $\mathbf{T}_3$). After reaching the altitude of about 120 m (event $\mathbf{E}_{18}$), a process of flap retraction from 8° to 4° (procedure $\mathbf{P}_3$) is commenced. Also, at this point the model changes the piloting task $\mathbf{T}_7$ to $\mathbf{T}_6$ in the attempt to maintain level flight (to increase the aircraft kinetic energy).

Note that this scenario diagram depicts, clearly and concisely, a difference between correct and incorrect control tactics at the level of cause-and-effect relationships between main events and processes constituting this complex enough flight case.

An example of autonomous simulation of flight according to scenario $\mathbf{S}_7$ is shown in Fig. 2.

Algorithm. A generic algorithm for executing a flight scenario in the situational model is as follows:

$$(\forall \mathbf{S})(\mathbf{S} = (\mathbf{E}_i, \mathbf{E}_k, \Pi_j)\ (((\mathbf{E}_i \in \Omega^P(\mathbf{E}) \wedge \mathbf{E}_k \notin \Omega^P(\mathbf{E}) \wedge \Pi_j \notin \Omega^{CL}(\Pi)) \wedge (t \geq t\,[\mathbf{E}_i \in \Omega^P(\mathbf{E})] + \tau)) \Rightarrow \Pi_j \in \Omega^A(\Pi)) \vee ((\mathbf{E}_k \in \Omega^P(\mathbf{E}) \Rightarrow \Pi_j \in \Omega^{CL}(\Pi)). \quad (20)$$

The relationship (20), together with the algorithm (16) which models flight events, and algorithms which model flight processes constitute a computational basis for the autonomous flight situation model.

Advantages. There are several advantages of using the autonomous flight situation model as a part of the hybrid flight simulator.

A flight scenario is a concise and clear mapping of key cause-and-effect relationships of flight and control in the form of directed graph. Complex flight situations, which are usually a dynamic superposition of several operational conditions, can be formalized in the most rigorous yet efficient way using the concepts of flight event, flight process, and flight scenario. A library of such scenarios can be constructed and retained for future reference or modification.

Verbal or other descriptions of various flight situations, such as Pilot's Manuals, flight accidents or incidents can be formalized and then modeled systematically using this model. By applying a "what-if" method, various actual, hypothetical and mixed flight situations can be constructed and analyzed directly by the pilot. Programming and piloting skills are not mandatory for using the model.

The situational model allows flexible planning and execution of a large volume of systematic flight training experiments. It can run faster or slower than the actual flight time. Finally, an important feature is that the complexity of flight situation planning and modeling does not depend on the complexity of a situation under study.

IMPLEMENTATION VARIANTS

There are several possible variants for implementing the hybrid flight simulator concept. The choice depends on training objectives, modeled aircraft types and funds available.

Fixed base control station. In this case a fixed base ground station is equipped with a control system to remotely control the UAV. Real-time images of out-of-the-cockpit views from the onboard camera are displayed on the Head-Mounted Display together with a computer generated image of the cockpit instrument panel. This option can be used, for example, for light airplane simulation.

Light-weight movable control station. This variant repeats the first one, but in this case the control station is installed on a six-degree-of-freedom dynamic platform. The main difference compared with a 6DOF ground based flight simulator is that there is no need to move the entire cockpit with its equipment which is very heavy. Only the pilot and a control panel is to be installed on the platform. In this case the weight of a loaded platform does not exceed 100-150 kg. This allows to use simple, energy efficient and relatively inexpensive dynamic platforms. The information obtained from the UAV attitude sensors is also used as command signals to drive the platform.

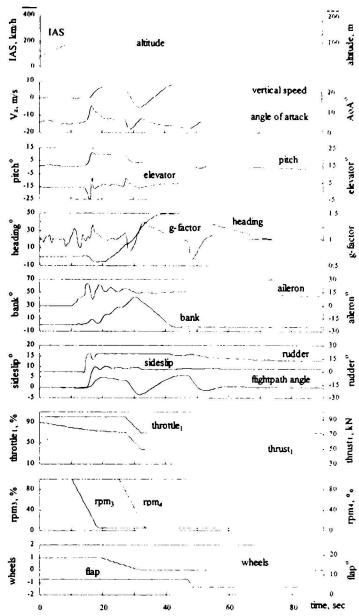


Fig. 2. Example of autonomous modeling of a flight situation S_7: "Takeoff with two right hand engines out"

A fixed base cockpit is used as the control station. In this option the pilot station is arranged as a fixed base aircraft simulator's cockpit. By means of a HUD or other virtual reality system, the pilot observes both a natural out-of-the-cockpit panorama and virtual controls located in the cockpit.

Partial use of ground based flight simulator hardware. In this case all the equipment of an existing ground based flight simulator can be employed, excluding the hardware which models the aircraft flight dynamics and the visualization system. Cockpit flight controls (e.g.: stick, pedals,

throttles, etc.) are used to control the UAV via a radio channel. The instrument panel imitators display the actual flight state parameters transmitted from the UAV. The data obtained from the UAV attitude sensors are simultaneously used as command signals to drive the platform.

DISCUSSION

In this section, several issues associated with concept implementation are briefly discussed.

One of the advantages of the hybrid simulator concept is a realistic portrayal of out-of-the-cockpit views. Also, this technology is compatible with existing ground based simulation complexes. The only components which should be changed are the visualization system and the flight dynamics modeling system. This allows a gradual and two-way transition from one configuration to another.

Modern telemetry and command systems can secure reliable communication with a UAV at a distance of 100 km and more. If communication satellites are used, this distance can be increased significantly. Modern UAVs have the maximum endurance between 4 and 10 hours. This allows to conduct training flights over remote territories located several hundred kilometers and farer from the base.

One of the pluses of this technology is the capability to naturally model various malfunctions and failures, including mechnical damages to the airframe and control surfaces.

The use of the autonomous flight situation model onboard and in the ground station can significantly expand the freedom of performing demonstrative and training scenarios. It also helps automate the process of flight scenario planning. The 'silicon pilot' model, which is a part of the situational model, supports various forms and methods of vehicle control, e.g.: recorded control tactics, autonomous scenarios (conducted by the model), and mixed control of the vehicle (by the model and the pilot), etc.

The supported training scenarios range from standard and non-standard flight situations described in the Pilot's Manual to flight accident and incident reconstructs and 'what-if' cases around some accident and various hypothetical maneuvers. In fact, the pilot can construct and retain a library of such scenarios for future reuse according to his (her) specific needs. Finally, the 'silicon pilot' model can be used as a backup flight control method for the UAV in case of a loss of communication between the vehicle and the ground station.

As UAVs are light vehicles they can be equipped with a parachute rescue system. This provides the flying model with a dual protection capability in emergencies (the first one is backup flight safety control of the situational model). Thus, if the student has made a piloting error, or has applied an incorrect recovery tactics, he (she) can observe realistic consequences of these actions without a fear to loose the vehicle. Also, the situational model can be used by the instructor to automate the process of planning and simulation of various internal and external operational factors.

Finally, in military aviation this technology opens a new avenue for sophisticated and realistic combat training. For example, air-to-surface attack scenarios have practically the same appearance as in reality, because all the targets may be real objects. This method also allows heterogeneous training scenarios which may involve multiple UAV and other systems (ships, tanks, etc.).

However, as any new technology this method may have potential problems and other unresolved issues which require a more thorough analysis.

One of the problems is how to integrate the external picture (out-of-the-cockpit view) recorded by onboard cameras and displayed on the HUD together with the cockpit's instrumental panel. Each of the implementation variants discussed above may require a special solution approach. Another problem is how to control these videocameras to account for dynamics of the pilot's head.

There is a problem of adequate modeling of takeoffs and landings and other flight regimes in close ground proximity. To secure adequate visual cues, the pilot's eyes (i.e. the onboard videocamera) should be located at the same level above the ground as in the modeled aircraft. However, the height of the UAV's undercarriage system is much lower than in actual aircraft. One of possible solutions could be to dynamically adjust the camera's optical focus distance to visually 'increase' it. Also, for these models of flight the takeoff and landing speeds should be the same as in modeled aircraft. This poses additional requirements to the UAV's undecarriage system, because originally it was designed for much lower takeoff and landing speeds.

If a dynamically similar [flying] model is used as the base vehicle, the problem is also how to expand the time scale without violating the dynamic similarity criteria. One of the approaches is to introduce into the DSM flight control system special devices to control its mass and inertial characteristics.

Several problems are to be addressed in order to integrate the situational flight model with the UAV and the human pilot. One of them is the identification of a set of the flight conditions (in an emergency or in case of loss of ground-vehicle communication) when the control authority should be transferred to the 'silicon pilot'. Another problem is visualization of the flight scenario dynamics in the ground station.

A special graphic interface program is also needed for construction, modification and selection of flight scenarios by the pilot/instructor. Finally, though memory requirements to run the situational model and retain a flight scenario are very modest (~50-100 K and ~5-10K, respectively), an onboard computer is required for this purpose. There are some technical problems associated with the implementation of joint (model-pilot) flight control.

Finally, one of possible drawbacks of this technology compared with the ground based flight simulation method is the necessity to actually fly the vehicle. In other words, a runway is required for takeoffs and landings together with a special zone in the air space to perform training flights. However, flying schools and training centers are normally located close to airfields.

CONCLUSION

For the last few years the cost of flight simulators for pilot training has increased significantly. The expansion of operational domains of flight of new aircraft requires pilots to receive more training at the edge of the flight envelope and under multi-factor conditions. As a result, pilot training is becoming less affordable than it should be and requires to address more complex flight situations.

The proposed technology of hybrid flight simulation is a feasible solution to these problems. The cost of flight simulation for pilot training can be reduced and the fidelity of flight modeling and the quality of pilot training can be improved. Some of the technical issues associated with the hybrid simulator concept require a more detailed analysis.

REFERENCES

1. "Trainers & Simulators", *Aviation Week & Space Technology*, December 2, 1996.
2. J.M. Rolfe, K.J. Staples, "Flight Simulation", Cambridge University Press, 1986.
3. Jeffery R. Hogue, R. Wade Allen, "Virtual reality entertainment technology applied to military parachute simulation flight training". Collection of Technical Papers of the AIAA Modeling and Simulation Technology Conference, New Orleans, LA, 1997, pp. 83-89.
4. L.M. Berestov, V.V. Gorin, "Simulation of flight dynamics by means of a flying laboratory", Moscow "Mashinostroenie", 1988 (in Russian).
5. Michael A. Dornheim, "X-36 To Test Agility of Tailless Design", *Aviation Week & Space Technology*, March 4, 1996, pp. 20-21.
6. Patuxen River, "NASA Drop Model Sinks", *Aviation Week & Space Technology*, January 20, 1997, p. 59.
7. S. Sadovnitchii, V. Ryabkov, A. Rushenko, J. Sandoval, "Modeling of Aircraft Flight by Means of Dynamically Similar Models with Flight Control System Similarity", *Proc. of AIAA, Modeling and Simulation Technology Conference'97*, New Orleans, 1997, pp. 326-332.
8. I.Y. Burdun, "An AI Situational Pilot Model for Real-Time Applications", *Proceedings of the 20th Congress of the International Council of the Aeronautical Sciences, Sorrento, Napoli, Italy, 8-13 September 1996 (ICAS'96)*, Volume 1, AIAA, 1996, pp. 210-237.
9. I.Y. Burdun and D.N. Mavris, "A Technique for Testing and Evaluation of Aircraft Flight Performance During Early Design Phases" (Paper No. 975541), *World Aviation Congress (WAC'97), October 13-16, 1997, Anaheim CA*, SAE Aerospace - AIAA, 1997, 12 pages.
10. P.G. Thomasson, "Flight Dynamics Simulation", Cranfield University, Cranfield, 1993.
11. R.W. Hamming, "Numerical Methods for Scientists and Engineers", McGraw-Hill, 1962.
12. V.V. Vyshinsky, "Aircraft Vortex Wake and Airport Capacity", (Paper No. 975519), *World Aviation Congress (WAC'97), October 13-16, 1997, Anaheim CA*, SAE Aerospace - AIAA, 1997, 13 pages.
13. Gamon M. A., "Airplane dynamic wheel loads during ground maneuvering", *AIAA/ASME/AHS 20th Struct., Struct. Dyn. and Mater. Conf.*, 1979, New York, pp. 95-101.

EXPERIMENTS AND PROBLEM ISSUES IN SIMULATING REAL-TIME PERSON-IN-LOOP AIR-TO-GROUND RADAR IMAGERY

Karl A. Spuhl
Associate Technical Fellow
and
Daniel A. Schmidt
Principal Technical Specialist
The Boeing Company
St. Louis Missouri

ABSTRACT

The objective of this paper is to present some of Boeing's experiments involved in generating radar databases and simulating radar imagery. The objective of these experiments was to improve image quality while reducing the cost of creating radar databases. This paper show's how undesirable imagery can result when visual OTW oriented databases are directly formatted into radar databases. Also presented are some experiments investigating ways to augment data extracted from visual databases with additional radar information. These experiments help focus attention on the need for the simulation community to work toward comprehensive database formats, standards, and architectures. Some of the experiments were successful in that they were able to function in real-time while others will require more investigation. This paper is an informational treatise focusing on individuals involved in the management and implementation of simulation technology. The attempt is to increase the awareness of the difficult issues involved in radar simulation. These issues, however, are also applicable to other types of sensor simulations.

INTRODUCTION

In many respects, simulating radar imagery is more difficult than simulating the visual out-the-window scene. Specifically it is more difficult simulating real-time person-in-the-loop radar imagery. These difficulties are the result of a number of significant interrelated issues. The issues include problems brought on by the lack of standards and communication within the simulation community as well as technical problems. Many of these problems are largely unnecessary, nevertheless they exist and we have to work around them. Everyone involved in radar simulation is addressing these problems independently and coming up with different solutions. This increases the cost of the simulator databases, the costs of interoperability, and the sharing of databases. One issue is the dominant position and influence visual out-the-window (OTW) simulation holds in the simulation industry. This has impeded the development of sensor simulation, including radar simulation. Another simulation community issue is that too few involved in the industry fully appreciate or understand the problems involved in simulating radar imagery. Clearly, the primary goal of a radar simulator is to generate high fidelity radar imagery and to have good correlation with visual and other sensor displays. An additional goal is to display the imagery in the same time interval as the real radar. We must solve many technical problems to meet these objectives.

Government and industry agree that a common data base architecture is needed. Acquiring agreement as to what the content of the architecture should be will be difficult. Achieving agreement in the future could be even more difficult as sensor technology advances and the tactical deployment of these systems becomes more sophisticated. These advances and applications will demand greater fidelity, require higher correlation with other systems, and further stresses the time budget in which to generate an image. These in turn will require documenting even more sophisticated and exacting information about the environment than is currently being modeled. This will make the database architecture more complex and expensive to produce. Boeing has been able to manage these issues primarily because their simulation thrusts were self-contained in-house applications. With the increased focus on interoperability among non-self-contained simulations, the problem becomes one the entire simulation

Copyright © 1999 by the Boeing Company.
Published by the American Institute of Aeronautics and Astronautics, Inc., with permission

community must address. The database issue is, in this writer's opinion, the biggest problem facing radar simulation today.

Computers are faster and more economical than they were ten years ago. These advances are certainly improving the ability to generate radar imagery in real-time. However we are still pushing the limits on how many radar artifacts, details, and anomalies we can process in the same amount of time that real radar's may take to produce an image. No matter how much computer speed we have, we seem to find a way to use it up to do "just a little bit more". In the area of correlation and interoperability, attention is again focused on the growing need for a source database architecture that supports both visual and sensor simulations. This includes visual, radar, infrared, and moving map displays. Visual system databases and sensor (radar) databases are often generated independent of one another. Most often visual databases are generated first. The database may or may not contain all the needed radar significant information. This visual database usually becomes the defining database for the simulation. This apparent separation of databases seems to be increasing with the growing popularity of simulation interoperability. In the environment of interoperability it is becoming increasingly common to work with third party generated databases. Often these databases do not support the unique requirements of the targeted radar simulator. The author hopes that the SEDRIS (Standard Environment Data Representation & Interchange Specification) effort will contribute toward a solution to this problem.

Industry, in general, focuses more time and resources on generating visual imagery than sensor imagery. Visual image generators dominate the market whereas sensor image generators tend to be more obscure and take second place. In the effort to develop a standard simulator database (Project 2851) and in the effort to develop a standard interchange format (SEDRIS), visual issues receive greater attention. This is understandable. The demand for visual systems is greater. It is easier and more natural to relate to visual imagery than to sensor imagery. Also there are more visual models than radar models available for purchase. Again, this is understandable. There is a greater market for visual models than radar models and visual models are easier to define. In addition, database generation tools are predominately focused on generating visual databases. This is not the case with radar models. The military and industry community cannot always agree on what is important or what needs to be simulated in the radar image. Therefore there is little financial incentive to take the initiative and market sensor-related database architectures.

FIDELITY AND CORRELATION

Achieving good fidelity in radar imagery is difficult. A major contributor is that the issues involved in generating high-fidelity radar imagery are not really understood by the simulation community. All too often the sensor community is forced to format a visual database into a radar database that really does not contain all the necessary information to simulate good radar imagery. This approach, for the most part, does not render high fidelity imagery. Consider how trees are often modeled in visual databases. One method uses several intersecting polygons as shown in Figure 1.

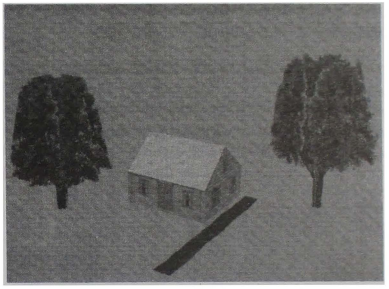


Figure 1: Trees Using Intersecting Polygons

Another method uses a single polygon with an image of a tree attached to the surface. In this method the polygon is continually rotated so that it always faces the viewer. Although both of these methods result in acceptable visual imagery they are unacceptable for simulating physics-based radar imagery. The data for the tree, in most cases, contains no radar specific information. Some texture data inherently contains a little information. However, writing software to handle the diversity in the ways textures are implemented, is costly and never ending. What is available is the geometry of the tree and hopefully some recognizable name that identifies the entity so that a reasonable guess can be made as how to estimate a radar return. Figure 2 illustrates models of trees containing all the information in the radar database that was derived directly from the information contained in the visual database.

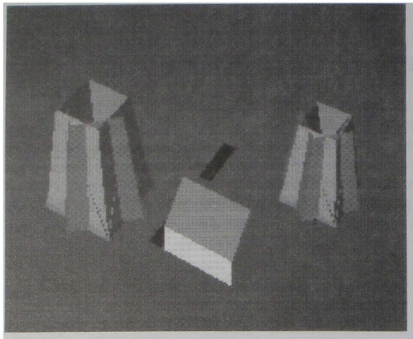


Figure 2: Radar Database Derived From Visual Database

This polygonal rendering of the tree geometry has its disadvantages. What occurs is that RF energy is striking uniform plane surface and numerous corner reflectors. The resulting imagery is shown in Figure 3 and is unacceptable. Note the stark rendering of the intersecting planes making up the two trees to the right and left of the house in the center.

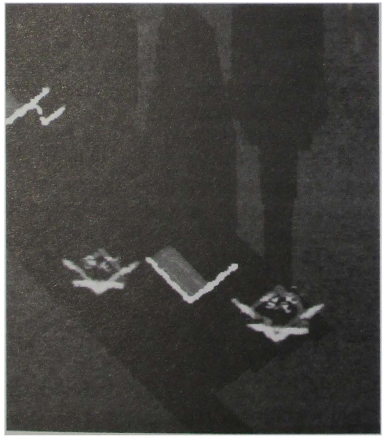


Figure 3: Radar Image Generated From Visual Database

Figure 4 represents an actual radar image of three trees rendered at one-foot resolution.

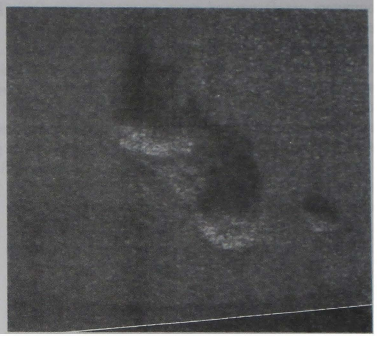


Figure 4: Actual Radar Image of Trees

Another example of poor radar imagery, resulting from formatting visual database information into a radar database, is transmission towers. As in the case of trees, photographs of transmission towers are often pasted onto polygons. This is illustrated in the left image of Figure 5. The right hand image is the corresponding radar database derived from the information contained in the visual database.

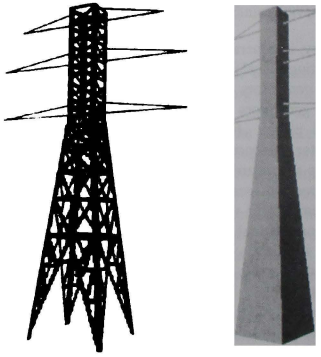


Figure 5: Left - Visual image of a transmission tower. Right - Radar database of the tower formatted from the visual model

Figure 6 illustrates a simulated radar image using the tower model derived from the visual model.

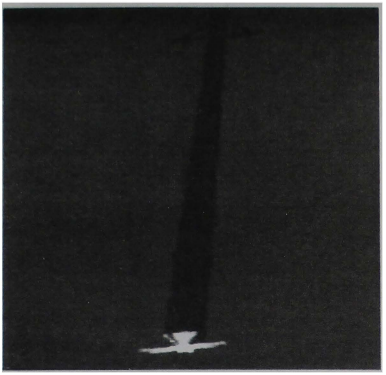
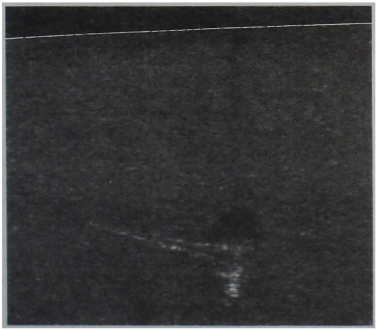


Figure 6: Simulated Radar Image using Visual Model Information

Figures 7 represents an actual radar image of a transmission tower rendered at one-foot resolution. Note the returns from the two transmission wires extending to the left of the tower. Also note the hemispherical shadow immediately behind the tower, and a slight hint of a shadow extending toward the top of the photo.



Figures 7: Radar Image of Actual Transmission Tower Showing Returns from Tower, Wires, and Slight Hint of Shadowing Top of Image

What the above examples illustrate are that visual databases, in general, do not contain sufficient information to support simulating high-quality radar imagery. It can be argued that correlation is exact since a tree and a transmission tower exist at the same locations and that the geometry of the entities is also identical. However, correlation goes beyond physical geometrical correlation. The definition of correlation should include how well the simulated image correlates with actual imagery. This could also be described as fidelity. Clearly the simulated imagery of the trees and transmission towers deviated considerably from actual radar imagery.

One solution, obviously, is that when an environmental database is generated, it includes all the information needed for simulating visual, radar, IR, moving map, and all other image types. This certainly is not a new idea, it has been discussed during Project 2851, the Distributed Interactive Simulation Workshops, the Simulation Interoperability Workshops, and in SEDRIS. It is not being implemented yet because the task is quite formidable. However, simulations still need to be done. Requirements for higher fidelity and better correlation continue to increase. In addition, everyone tries their best to deliver simulations as expeditiously and economically as they can in this imperfect environment.

MODEL SUBSTITUTION EXPERIMENT

One of the experiments Boeing conducted in order to improve the appearance and fidelity of radar imagery involved a two-part effort. The first part consisted of constructing high fidelity radar models of visual entities whose databases did not provide suitable information to render acceptable radar imagery. The second part of the experiment investigated automatically substituting each occurrence of a visual model with the higher fidelity radar model when the visual database was being formatted into a runtime radar database. Figure 8 shows an example of a high fidelity radar model of a tree. This particular tree model was constructed to investigate the radar imagery that would result if a large number of leaves were processed.

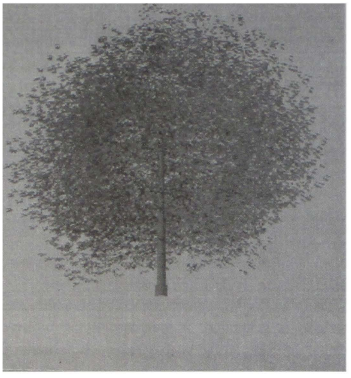


Figure 8: High Fidelity Radar Models of Trees and Transmission Tower. Right – Tower detail

In a similar experiment, transmission tower models were also evaluated. Figure 9 is a visualization of a high fidelity radar database of a transmission tower. In

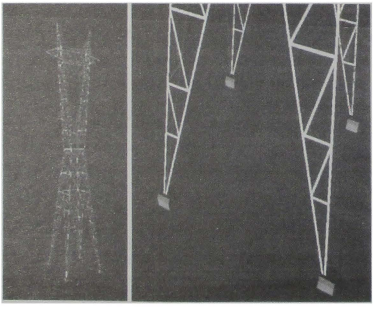


Figure 9: Visual rendering of the Transmission Tower's Polygonal Configuration. At Right, a Detail of the Tower Base

this database each individual structural beam is rendered as a steel polygon. The tower rests on concrete pylons. Figure 10 illustrates the resulting radar imagery using the higher fidelity tree and transmission tower models.

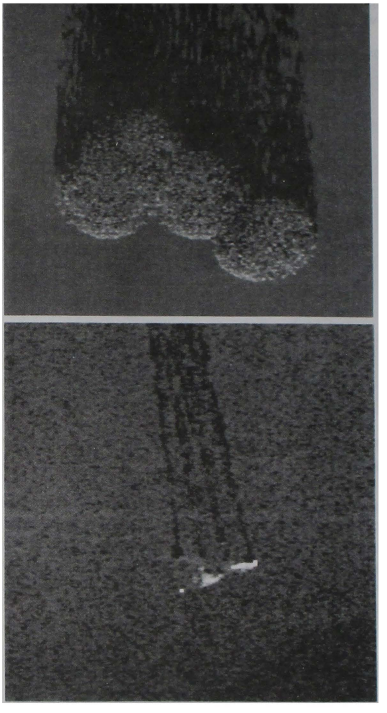


Figure 10: Simulated Radar using higher Fidelity Tree and Transmission Tower Radar Models

The substitution process was an improvement, however several difficulties were encountered. The visual database provided information on the location of the transmission towers but none on their orientation. To fix the problem it was necessary to calculate the "path" of the towers along the ground and then orient the towers so the "arms" were perpendicular to the path. For the most part this worked. It would have been a lot easier and less costly had the visual database initially contained the orientation information. The process did require that you knew which radar towers to use to replace different tower shapes in the database. A cross-reference table unique to the database had to be generated.

Substituting trees was more difficult. There was no consistent information as to what size, shape, and type of trees a model represented. Also the approach used to model the trees still requires more investigation. The models produced excellent radar imagery but the trees took so long to process that real-time operation was no longer maintained.

SEPARATE ENTITY DATABASES

It is not always practical to model an entire fly-space at the highest resolution required. This is costly to produce and uses enormous quantities of disk space. What is often done is to model important target areas, turn-points, and corridors to the needed level of resolution and then model the remaining fly-space at a lower but acceptable resolution. However the aircrew can point their antenna anywhere in the fly-space, and usually do, and request imagery at any level of resolution. The actual database implementation we use is shown in Figure 11.

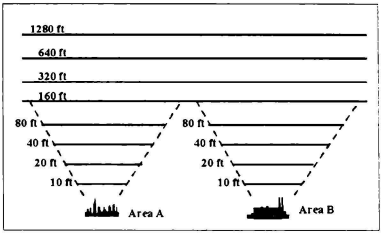


Figure 11: Database Resolution Implementation

Note that areas A and B are modeled at, say 10 ft per database element (pixel). The next level is modeled at 20 ft or twice the resolution as the first level. It also covers a larger area. The next level is modeled at 40-ft resolution and covers a larger area than the 20-ft resolution level. This inverted cone pattern continues to roughly 160-foot resolution. At that point, the entire database is modeled at 160, 320, 640 and 1280 foot resolution. What this accomplishes is that the aircrew can generate low-resolution imagery anywhere in the fly space to help locate their turn-points and target areas. Once the general location of the target area is found, the aircrew can go to higher and higher resolution images in the target region. The aircrew, however, often attempts to generate high-resolution imagery where high-resolution data does not exist. Figure 12 illustrates the resulting imagery.

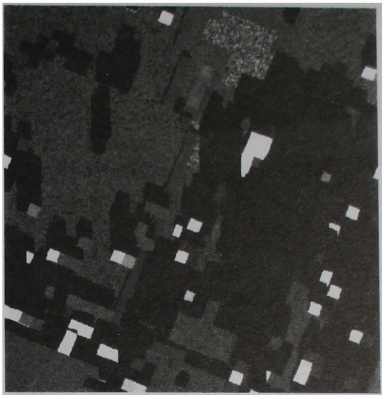


Figure 12: Attempt to Generate High-Resolution Imagery from Low-Resolution Database

We wanted a way of rendering high resolution appearing radar imagery in these areas while limiting the amount of data storage required. It was suggested that we remove all entities resting on top of the terrain skin, render them at all levels of resolution, and place them in a separate file (Patent Pending). The term entity, as used in this paper, refers to any feature that rests on the terrain surface and has some height associated with it. This includes all buildings, trees, and vehicles. The terrain skin database would only contain references as to where the entities were located. Then, during runtime, the entities would be placed onto the terrain surface. Thus if the radar was directed to process a higher resolution image than the terrain database supported the simulator would insert higher resolution models into the scene. This approach worked well and we were able to maintain real-time operation. Furthermore, the size of the database defining all levels of resolution for the models did not use as much disk space as expected. Figure 13 shows a high-resolution model substituted for what would have otherwise been a low-resolution version.

There was a downside to this approach and it is also illustrated in Figure 13. The terrain database, which includes only elevation and area features such as roads, rivers, etc., was not modeled at the resolution levels needed to support the requested level of fidelity. Using low-resolution features to generate high-resolution



Figure 13: High resolution Model Substituted in Place of a Low Resolution Mode.

scenes' results in the low-resolution features exhibiting alias effects.

SELECTIVE ENTITY PLACEMENT

The above approach resulted in several additional advantages. Since it was determined that all our entities could be stored in a separate library and placed in the terrain during run-time, we experimented with the idea of allowing the user the choice to place or not place any or all entities. It requires the user to identify those entities off line before the simulations are run. This worked excellent.

We went further and experimented with the idea of allowing the user to reposition any entity anywhere in the database. This also worked but there was a limitation. To maintain real-time operation we needed to define two types of entities. We call them dynamic and static entities. Dynamic entities are models that can be selected from the library, located, and relocated anywhere in the database during runtime. We determined that we could accommodate 5000 dynamic models. Static entities are models that can also be relocated with the following limitations, their rotation is fixed (static) and they must be identified off line before the simulation is run. The reason not all entities were made dynamic is that we would not be able to maintain real-time operation with that large number of dynamic models. It takes more time to process dynamic models. There is one other limitation. There is not enough bandwidth between the simulation host computer and the radar computer to relocate all 5000 dynamic models each cycle time. These capabilities allow the user to configure and fine-tune the content of the run-time database to suit specific training needs. It provides the capability of producing moving vehicles over terrain and water.

INTERVISIBILITY-CORRELATION AND FIDELITY

Visual databases primarily use polygons to model terrain. Radar imagery rendered from polygonal surfaces tends to produce uniform sterile returns and unrealistic imagery. Although radar simulators can support high-resolution gridded databases, polygons are often required to maintain intervisibility correlation. Figure 14 illustrates intervisibility correlation.

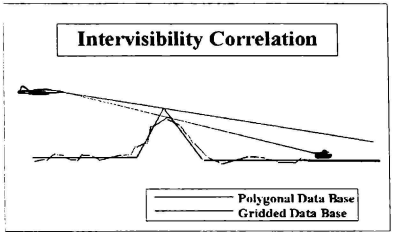


Figure 14: Intervisibility Correlation

Consider the scenario where the radar simulator is using high-resolution gridded elevation. The aircrew could locate a potential target over the top of a hill. If then the target location information is handed off to a FLIR using a polygonal elevation database, the FLIR could be looking at the side of the hill. This is why radar simulators often use polygonal elevation.

We experimented with the idea of improving the appearance of the polygonal terrain while maintaining exact intervisibility correlation. It essentially required the use of two elevation databases. Figure 15 illustrates radar imagery generated using gridded elevation. In comparison Figure 16 illustrates radar imagery generated entirely from polygonal elevation. To improve the appearance of the polygonal image and still maintain intervisibility correlation, we first computed shadows using the polygonal elevation. Then we computed the normal vector at each polygonal grid element based on the gridded elevation (Patent Pending). This normal vector is used to calculate the incident angles between the radar transmitter and the

terrain surface. What this accomplishes is that it breaks up the otherwise uniform return you would get from a plane surface. This is illustrated in Figure 17. Note that the shadowing is identical with Figure 16, hence visibility correlation, while the appearance closely resembles the higher resolution gridded image of Figure 15. This processing was successful in maintaining real-time operation.

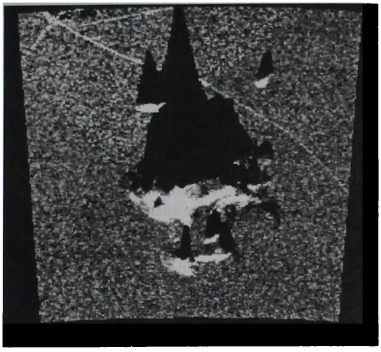


Figure 14: Image Generated form Gridded Elevation



Figure 15: Radar Image Generated from Polygonal Elevation

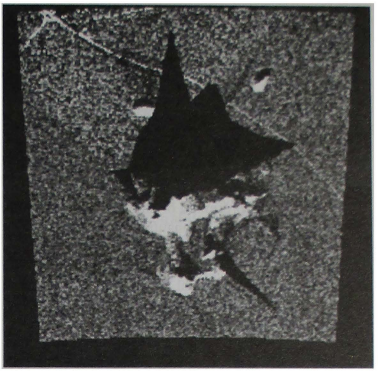


Figure 16: Radar Image Generated from Polygonal Elevation with Incident Angles Calculated form Gridded Elevation Data

CONCLUSIONS

In order to improve simulation interoperability, the simulation community needs to work together and develop a comprehensive source database architecture that supports all types of imagery needs. The concept underlying the SEDRIS Project is an opportunity to help meet these needs. Identifying the types of information to be documented in a comprehensive source database will help educate and enlighten the simulation community to the needs of sensor simulation. It would also provide direction to developers of database generation software to provide the necessary tools to help support sensor databases.

A considerable amount of unnecessary effort and resources are being used in interrogating third party databases (mostly visual) for radar significant information and supplementing missing radar information. Although visual databases contain a significant amount of information helpful in simulating radar imagery, it is still largely deficient in the types of information needed for simulating high fidelity imagery.

An Overview of Atmospheric Modeling for Aeronautical and Aerospace Vehicles Simulation Applications

William W. Vaughan, University of Alabama in Huntsville, Huntsville, AL
Dale L. Johnson, NASA Marshall Space Flight Center, Huntsville, AL
L. J. Ehernberger, NASA Dryden Flight Research Center, Edwards, CA

ABSTRACT

An overview is presented of atmospheric modeling relative to aeronautical and aerospace vehicles design, flight simulation, and mission operation applications. Included is a review of the various environmental phenomena and areas of design and mission assessment concern. In particular, the paper discusses the sources of measurements, modeling issues, design application philosophies and examples of models developed for use by the engineering community. This review is based on many years of experience associated with atmospheric model developments and applications to various aeronautical and aerospace programs.

1. Background

The terrestrial environment is an important forcing function in the design and development of aeronautical and aerospace vehicles. The scope of the terrestrial environment includes the following phenomena: ground and in-flight wind and turbulence models; atmospheric thermodynamic models and properties; thermal radiation; humidity; precipitation, and icing; fog, cloud characteristics and cloud cover models; atmospheric electricity; atmospheric constituents; vehicle engine exhaust and toxic chemical release; U.S. and World surface environment extremes; occurrences of severe weather including tornadoes and hurricanes; geological hazards; and sea states. Since the flight profile of any aircraft or spacecraft vehicle is part or totally within the terrestrial atmosphere, its operation will always be influenced to some degree by the terrestrial environment with which it interacts. As a result, the definition of the terrestrial environment and its interpretation is one of the key vehicle design and development inputs for the various engineering systems.

Copyright @ 1999 American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty free license to exercise all rights under the copyright claimed herein for Government purposes. All other rights are reserved by the copyright owner.

Although the near-earth terrestrial environment is the major environmental driver for aeronautical and aerospace vehicles, the natural environment above 90-km altitude must also be considered for design of aerospace vehicles. The orbital phase of an advanced aerospace vehicle includes exposure to space environment phenomena such as atomic oxygen, atmospheric density, ionizing radiation, plasma, magnetic fields, meteoroids, orbital debris, etc. Specific aerospace vehicle terrestrial and space environment requirements are normally specified in the appropriate vehicle design ground rules and criteria documentation.

It is, therefore, important to recognize the need for the definition of the terrestrial environment very early in the design and development cycle of any new aeronautical or aerospace vehicle. Using desired operational capabilities and flight profiles, specific definitions of the terrestrial environment can be developed and provided which, if the vehicle is designed to accommodate, will ensure the desired operational capability within the accepted risk level. It is very important that those responsible for the terrestrial environment definition have a close working relationship with both program management and design engineering personnel, to ensure that the desired operational capabilities are reflected in the terrestrial environment requirements specified for design of the vehicle.

A vehicle's response to terrestrial environment design criteria must be carefully evaluated to ensure an acceptable design relative to desired operational requirements. The choice of criteria depends upon the specific operational locations(s), vehicle configuration, and expected trajectory and mission profile and mission frequency. Vehicle design, operation, and flight procedures can be separated into particular categories for proper assessment of

environmental influences and impact upon the life history of each vehicle and all associated systems. These include categories such as: (1) initial purpose and concept of the vehicle, (2) preliminary engineering design, (3) structural design, (4) control system design, (5) flight mechanics, orbital mechanics and performance (trajectory shaping), (6) aerodynamic heating, (7) takeoff/landing capabilities, (8) optimization of design limits regarding the various and/or more critical environmental factors, and (9) final assessment of environmental capability for safe and reliable flight operations.

2. Significance of Wind Inputs

Aeronautical Vehicles and Wind Environment:

Weather elements have impacted aircraft since the very first flight attempts. The first technical report produced by the National Committee for Aeronautics (NACA) dealt with the flying qualities necessary for control in irregular wind conditions. Solutions to these early wind problems involved improved aerodynamic configurations and control effectiveness and more precise descriptions of the wind dynamics or statistical characteristics. As aircraft entered routine service in the military and commercial applications, design considerations for maneuver loads were joined by concerns for the maximum vertical gust load and airframe fatigue due to the accumulated effects of repeated exposure to turbulence. Airplane center-of-gravity normal acceleration data were recorded and analyzed during the era beginning in the 1930's. These normal acceleration data were interpreted as the vertical gust velocity that would cause the measured gust load increment by increasing the airplane angle-of-attack. Statistical compilations of the gust survey data characterized the natural environment gust load inputs as a function of altitude for several geographic areas and types of aircraft operations. By the 1950's and 1960's the "derived equivalent gust velocity" concept matured with the specification of a "one-minus-cosine" shape for civil and military design requirements.

In addition to gust loads, design requirements also called for assessment of airplane control and flying qualities in turbulence. Design requirements (certification procedures) called for evaluation of airplane response to a sinusoidal vertical gust input with frequency or wavelength varied over a broad range from negligibly short gusts to those so long that airplane response would essentially be steady state. Adverse responses found at any frequency or wavelength must be remedied or mitigated by aerodynamic or controls modifications.

During the 1960's, true gust velocity data measured by several specially instrumented aircraft were used to define random turbulence models specified by the "Dryden" and "von Karman" power-spectral-density (PSD) formulations. Scale, intensity and amount of turbulence (expected/encountered) parameters for the Dryden and von Karman PSD design criteria were based on matching the PSD model loads with those documented by the derived (discrete) equivalent gust survey database within each altitude layer. The resulting PSD models became design tools for both aeronautical and aerospace vehicles. Later in the 1970's and 1980's, PSD criteria became sufficient for passenger transport aircraft certification but aerospace vehicle design maintained a reliance on wind criteria such as the synthetic wind profiles, vector wind profile descriptions, and multi-mission wind profiles analyses.

Wind Environment Interactions Relative to Aerospace Vehicle Design:

Wind is usually the most important terrestrial environment parameter influencing the design of an aerospace vehicle. Because it has temporal and spatial variations, representation of the wind data in a simple and concise form is not possible for all purposes of design and operation of the vehicle. Caution must be exercised in the employment of wind data to ensure consistency with the physical interpretation relative to the specific vehicle design problem. Given the importance of wind inputs for vehicle design, wind input has been selected to illustrate the importance of terrestrial environment considerations in the vehicle design and development process.

a. Ground Winds

Since wind is a vector quality that varies in both space and time, wind loads should be considered as a dynamic input to the vehicle structure. From the aerodynamic point of view, it involves viscous separated flow around a blunt body. Such flows have been studied for many years, but there does not exist an adequate approximation to aerodynamic transfer functions

to relate the dynamic wind vector to the dynamic load on the vehicle. Therefore, quasi-steady assumptions must be used. The near-surface wind profile is broken down into steady and unsteady components, with the resulting static and dynamic loads superposed. The second order loads, which result from the interaction of the static and dynamic wind components, are often neglected in the present state-of-art.

b. In-flight Winds

A vehicle derivative used to launch a new payload location configuration further exemplifies a wind gust elastic body response-coupling problem. This small change in external payload configuration can change the aerodynamic distribution, thus the response to winds, and have a substantial effect on the resulting loads. For example, a vehicle configuration had approximately a 95 percent launch probability without wind biasing, and greater than 99 percent with wind biasing. This assessment was made using design synthetic wind profiles and verification and operational analysis using a Monte Carlo approach with the 150 per month measured Jimsphere wind profile ensembles. With the new payload, the vehicle configuration had less than a 50 percent launch probability without wind biasing and 80 percent with wind biasing using the synthetic profiles. As a result, the problem was resolved using wind biasing and verified using Monte Carlo response analysis. The lesson is clear, small vehicle changes can easily eat up large margins. Analysis must be refined to match the engineering problem requirements. Two key vehicle-engineering systems are now considered.

Guidance System

The ascent wind profile effects come into play when the guidance system attempts to meet the objective of maximizing payload for a trajectory through the Earth's atmosphere. One factor, on which the wind has a direct influence, is the effect of flight path on vehicle bending moment and hence on structural weight. Bending moment is caused by control forces, i.e., engine gimbaling, and by aerodynamic side forces induced by wind action against the side of the vehicle and vehicle maneuvering. The size of bending moments on a vehicle determines in part the structural strength requirements and, thus, the structural weight of the vehicle. With all other factors considered equal, higher structural weight results in lower payload. Therefore, in choosing the optimal flight path, the guidance system must consider not only the flight-mechanical aspects, but also the wind-induced bending moments, rigid body aerodynamics, aerodynamic lift and vehicle overturning moment contribution to the drift and, thus, flight path optimization. The wind environment definition plays and important part in this analysis.

Control System

The most significant terrestrial environment element affecting control system design is upper altitude wind in the high dynamic pressure region (10-15 km altitude). The control system functions primarily to maintain a prescribed flight path as generated by guidance on pre-programmed attitude tilt commands. Off-nominal values of vehicle parameters and the presence of winds will cause the flight path to differ from that anticipated by the guidance system. Ideally, the control system should minimize this difference. However, there is a cost incurred in attempting to respond precisely to the guidance commands, and the cost appears in terms of bending moments and resulting structural loading on the vehicle plus drift and later thermal loads when the vehicle corrects for flight path deviations. The fact that winds are acting on the vehicle could make this cost excessive. Winds aloft are frequently of such a large magnitude that large dispersions in guidance system prescribed attitude and flight path angles occur. In order for the control system to decrease these dispersions, large bending moments are imposed on the vehicle. If a controller is being designed for an already designed vehicle structure, these loads can be so large that the vehicle would exceed its design loads and break up. On the other hand, if the control system design is for a vehicle in the preliminary design stage so the structural requirements are yet to be established the large bending loads can result in excessively complex, heavy, or expensive structural configurations. Consequently, because of the in-flight winds, bending moments on the vehicle become the overriding consideration in controller design.

c. Summary Remarks

The determination of an aeronautical or aerospace vehicle's response to wind disturbances cannot be reduced to the evaluation of one discrete set of response criteria, such as

vehicle loads, but must include many response parameters. The parameters that become design drivers depend upon the vehicle configuration and specific mission profile. It is not practical to use only one design method for all phases of vehicle design. Different methods of evaluation must be used as the particular design phase demands. The phases include preliminary design, final structural design, guidance and control system design and optimization (preliminary and final), and establishment of limits (constraints) and procedures for takeoff/landing and flight operations. The definition and interpretation of the wind environment plays a critical role in all of these phases.

3. Wind, Wind Shear, Turbulence, and Temperature Gradients

Wind and wind changes impact many design goals and operational constraints such as: range, navigation, vehicle loads, climb performance, altitude hold handling qualities, etc. Upper-air steady state wind directions and speeds (or easterly and northerly wind vector components) statistics have been complied in numerous forms. Range reference atmosphere data for the primary launch sites and test ranges are tabulated by month giving mean values and standard deviations for the wind vector component and speed. Tabulated values for each km in altitude include correlation coefficients between the easterly and northerly wind vector components to define the probability ellipses are given. Statistical coverage for all altitudes over the whole hemisphere is incorporated into NASA's Global Reference Atmospheric Model (GRAM) applications software. Within this model winds can be simulated for each month of the year for user specified trajectories. In addition, the user can also specify statistical variations to represent variations encountered with changing weather patterns.

a. Wind Shear and Turbulence

Wind speed or vector component changes with distance are termed wind shear. Vertical wind shear in units of speed change per unit altitude are often expressed in inverse seconds (i.e. per second) over specified scales-of-distance for convenience in engineering design applications. Horizontal wind shears tend to be approximately two orders of magnitude less than vertical wind shears and, for aviation meteorology applications, are often expressed in units of knots per hundred nautical miles.

Measured wind-shear magnitudes depend on the depth of the layer over which the shear is measured. Detailed wind-shear observations and their statistical analyses for altitudes below about 60,000 ft have been abundant since the initiation of the manned space program. These observations have used special balloons (Jimsphere) tracked by high precision radar to altitudes near 60,000 ft, and conventional weather balloons to altitudes of about 100,000 ft. Jimsphere wind data studies accomplished by the NASA have provided a wealth of information on wind changes and wind shears and on the relationship between wind shear and the thickness of the altitude layer over which the shear is measured. Wind shear is measured to reasonable accuracy over layers thicker than about 200 ft by the Jimsphere and over layers greater than about 2000 ft by rawinsonde balloons used for routine upper-air weather observations. To bracket a range of stronger shear values, figure 1 illustrates a 90-percentile curve for July, when wind shears are weaker, and a 99-percentile curve for February when they are stronger. Wind shears exceed the values shown by these curves for 10 and 1 percent, respectively, of the observations.

Wind shears measured over large altitude distances (or thickness scales) have a positive correlation with wind speed and therefore tend to decrease from a maximum near jet stream altitudes to minimum shear magnitudes in the middle stratosphere, just above supersonic cruise altitudes. This decrease in wind shear magnitudes in the lower stratosphere is depicted in figure 2 by curves extracted from rawinsonde data for altitudes of 45,000 and 80,000 ft.

Turbulence models for airframe and aerospace vehicle dynamic structural response and controls flightworthiness assessment are specified by Federal Aeronautical Regulations, Military Specifications, and specific aerospace vehicle natural environment design requirements guideline documents. Lower altitude wind shear and turbulence models are applied to assure flight worthy aircraft performance during the takeoff, climb, and landing approach. Specific models are recommended for particular purposes such as "autoland" use during low visibility approach (Category II and III landing criteria)

and onboard wind shear detection system demonstration. For these applications specific shear values are specified. Extreme values of shear and turbulence due to severe storm downbursts and outflow have been observed and included in recent model updates. Model evolution has been a gradual process and continues as new applications and operation environments emerge (e.g., rotorcraft, supersonic and higher altitude observation vehicles, etc.).

b. Stratospheric Temperature Gradient Statistics

Because the speed of sound in air is a direct function of ambient temperature, the sonic boom propagation direction will bend when strong temperature gradients are traversed at an oblique angle. The strongest gradients are associated with inversion layers in the lower stratosphere. Statistical analyses of temperature gradient features have not been nearly as adequate as for wind speed and wind shear. There have been only a few reports which have presented the results of analyses of a limited amount of rawinsonde data for (a) strong lapse rates (temperature decrease with altitude) or (b) strong inversion rates (temperature increase with altitude) within layers surrounding the mandatory meteorological reporting levels. Such studies are motivated to improve the knowledge of probabilities for features associated with high-altitude turbulence experienced by supersonic cruise aircraft. Values depiciting 1.0, 0.1 and 0.01 percentile extremes for these data are shown in figure 3.

Note that in the lower stratosphere the magnitudes for the decreasing gradients are not as large as those for the increasing temperature gradients. At both the 0.1 and 0.01 percentile frequency-of-occurrence the decreasing temperature with altitude exceeds the adiabatic lapse rate (3 deg C/1000 ft), presumably as a result of measurement error in some cases and strong atmospheric dynamics in others. The increasing temperature gradient at 0.01 percentile rate, 24 deg C/1000 ft, is comparable to a density departure rate of approximately 10 percent/1000 ft from the standard day atmosphere, as well as to the speed of sound changing by 5 percent/1000 ft from the standard atmosphere values.

c. Summary Remarks

A hallmark publication by NASA, "A Compendium of Wind Statistics and Models for the NASA Space Shuttle and Other Aerospace Vehicle Programs" reviews 40 years of wind model development. This review covers the specification of discrete longitudinal gust magnitude as function of altitude and turbulence intensity risk probabilities. Also presented is a discussion on one of the key developments in wind profile representations for vehicle design; the vector wind model. This publication, along with NASA's publication "Terrestrial Environment (Climate) Criteria Guidelines for Use in Aerospace Vehicle Development", provides excellent sources for information on ground and in-flight regions' winds, wind shear, turbulence for use in aeronautical and aerospace vehicles design and development studies.

4. Thermodynamic Modeling

The surface and in-flight atmospheric thermodynamic parameters (temperature, pressure, and density) are used in (1) research planning and engineering design of remote sensing systems, (2) aerospace vehicle design and development and (3) trajectory analysis, dealing with vehicle thrust, dynamic pressure, aerodynamic drag, aerodynamic heating, vibration, structural and guidance limitations, and reentry analysis. A statistical analysis of these atmospheric parameters, in terms of means, extremes and probability risk levels, are needed to be applied to the various vehicle-engineering systems.

Site-specific Range and Standard atmospheric models have been constructed for essentially all the major test ranges in the United States. These provide monthly mean vertical profiles and their standard deviations for the various thermodynamic parameters from surface to mesospheric altitudes. NASA has further developed and refined a Global Reference Atmospheric Model (GRAM) which presents the monthly mean and standard deviations of thermodynamic parameters and winds on a global basis from surface through orbital altitudes. It is composed of the Global upper Air climatic Atlas data base from surface to 27 km altitude, the Middle Atmosphere Program data base from 27 to 90- km altitude, and the NASA Marshall Engineering Thermosphere Model from 90 to 2500 km altitude. GRAM can be output along a vertical profile, or along a given trajectory path for an aerospace vehicle. It also

has the capability of generating individual, realistic atmospheric profiles in a random number generated Monte Carlo sense. Thus, hundreds of different atmospheres can be generated along a given trajectory and then used in the various engineering analyses. GRAM was initially developed for the Space Shuttle design studies and is now used mainly in mission planning and in vehicle de-orbit, return, re-entry, and landing studies. GRAM was also baselined for the National AeroSpace Plane (NASP) program as the atmospheric model for use in design, development, and operational planning. A real-time GRAM simulation program was developed at NASA-Dryden for use in NASP pilot interaction and training studies and proved very successful. GRAM was thus modified in order to run in this special type of real-time flight simulator operating mode.

5. Summary

Terrestrial environment models and design requirements must be formulated and interpreted in terms of an aeronautical or aerospace vehicle's characteristics and operational requirements. This can only be accomplished with the terrestrial environment being embraced as part of the overall requirements for the vehicle system design to achieve an optimum design. True optimization is a balancing act between trajectory/flight path (performance) control, loads, thermal and operations where terrestrial environments are the major environmental drivers. A vehicle's design must consider thermal, control, performance (trajectory shaping), structural, electronic, materials, flight mechanics, etc. It must also include terrestrial environment definitions to provide a total systems approach.

Lessons learned show that it is very beneficial to establish the terrestrial environment design requirements early in the design process to avoid time consuming and costly redesign as later stages of development. During the very early stages of the design definition or a new vehicle, desired operational characteristics are developed. Given that the vehicle will operate in the terrestrial environment, design consideration must be given to the operational environmental conditions. Assessments made early in the development program will prove advantageous in maintaining an economical program and obtaining a vehicle with acceptable operational sensitivity/robustness to the environment. Also, for those terrestrial environment parameters that need accurate and timely monitoring prior to or during operations of the vehicle, early assessments will permit development of any necessary new environment measurement system(s) to support tests and operations.

In many cases, it is impossible to clearly define limiting extreme values for a particular terrestrial environment parameter that may occur during the desired lifetime of the vehicle. It may not be technically or economically feasible to design a vehicle to withstand an extreme environment value. However, a lower value may be defined whereby the probability is acceptably small that the lower value will occur during the desired lifetime for the vehicle. Because of these and other considerations, a value less than the extreme may be a more appropriate design input. The terrestrial environment specialist has the responsibility to provide the program manager and chief engineer pertinent information so they can determine the highest risk design value that is acceptable for the program in that particular environment areas. Therefore, it is very important that the program manager and chief engineer have a good understanding of the operational risks due to the terrestrial environment.

Bibliography

Anon. (1998): "Guide to Reference and Standard Atmospheric Models", ANSI/AIAA G-003-1998, American Institute of Aeronautics and Astronautics.

Arbuckle, P.D.; Lewis, M.S. and Hinton, D.A. (1996): "Airborne Systems Technology Application to the Windshear Threat", 20th Congress of the International Council of the Aeronautical Sciences, Sorrento, Italy, ICAS Paper No. 96-5.7.1, pp.1640-1650.

Barr, Neal M.; Gangsaas, Dagrinn and Schaeffer, Dwight R. (1974): "Wind Models for Flight Simulator Certification of Landing and Approach Guidance and Control Systems", FAA-RD-74-206.

Ehernberger, L. J. and Gutman, Nathaniel B.: "Climatological Characteristics of High Altitude Wind Shear and Lapse Rate Layers". NASA TM-813563.

Houbolt, J.C.; Steiner, R. and Pratt, K.G. (1964): "Dynamic Response of Airplanes to Atmospheric Turbulence Including Flight Data on Input and Response." NASA TR R-199.

Johnson, D. L., Editor (1993): "Terrestrial Environment (Climatic) Criteria Guidelines for Use in Aerospace Vehicle Development", NASA TM-4511.

Johnson, Dale L., Steven D. Pearson, William W. Vaughan and G. W. Batts (1997): "Role of Aerospace Meteorology in the Design, Development, and Operations of New Advanced Launch Vehicles", Proceedings of the 7th Conference on Aviation, Range, and Aerospace Meteorology, American Meteorological Society, Boston, MA.

Johnson, Dale L., Steven D. Pearson, William W. Vaughan and G. W. Batts (1998): "Assessment Regarding Impact of Atmospheric Conditions on Shuttle Launch Delays", 36th Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Paper AIAA 98-0293.

Johnson, D. L., C. K. Hill, W.W. Vaughan, S. C. Brown and G. W. Batts (1993): "Natural Environment Requirements Definition and Significance for Aerospace Plane Development", American Institute of Aeronautics and Astronautics, Paper AIAA 93-5075.

Justus, C. G., W. R. Jeffries III, S. P. Yung and D. L. Johnson (1995): "The NASA/MSFC Global Reference Atmospheric Model – 1995 Version (GRAM-95)", NASA TM-4715.

McGhee, D. S. (1999): "A Strategy for Integrating a Large Finite Element Model Using MSC NASTRAN (PATRAN: X-33 Lessons Learned", NASA TM-1999-209201.

Pearson, Steven D., Bonnie F. James, and William W. Vaughan (1995): "From Red Tape to Results: A Systems Engineering Approach to Natural Space Environment Challenges", 33rd Aerospace Sciences Meeting and Exhibit. American Institute of Aeronautics and Astronautics, Paper AIAA 95-0844.

Pearson, S. D., W. W. Vaughan, G. W. Batts and G. L. Jasper (1996): "Importance of the National Terrestrial Environment with Regard to Advanced Launch Vehicle Design and Development", NASA TM-108511.

Pearson, Steven D., Gwenevere Jasper, William W. Vaughan and Glen W. Batts (1997): "Natural Terrestrial Environment Importance to Advanced Launch Vehicle Design and Development", 35th Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, Paper.

Pratt, K.G. and Walker, W.G. (1954): "A Revised Gust-Load Formula and a Re-Evaluation of V-G Data Taken on Civil Transport Airplanes From 1933 to 1950", NACA-1206.

Range Commanders Council – Secretariat (1983): "Cape Canaveral, Florida – Range Reference Atmosphere 0 to 70-km Altitude", RCC Document 361-83.

Ryan, R. S. (1992): "The Role of Failure /Problems in Engineering: A commentary on Failures Experienced – Lessons Learned", NASA TP-3213.

Ryan, R. S. (1995): "Problems Experienced and Envisioned for Dynamical Physical Systems", NASA TP-2508.

Scoggins, James R. (1964), "Aerodynamics of Spherical Balloon Wind Sensors in Turbulent Flows", J. Geophysical Research, vol. 69, no. 4.

Smith, O.E., and S.I. Adelfang (1998): "A Compendium of Wind Statistics and Models for the NASA Space Shuttle and Other Aerospace Vehicle Programs", NASA CR-1998-208859.

Switzer, G.F.; Proctor, F.H.; Hinton, D.A. and Aanstoos, J.V. (1993): "Windshear Database for Forward-Looking Systems Certification". NASA TM-109012.

Fichtl, George H. (1971): "The Responses of Balloon and Falling Sphere Wind Sensors in Turbulent Flows", NASA TN D-7049.

Vaughan, William W. (1977): "An Investigation of the Temporal Character of Mesoscale Perturbations in the Troposphere and Stratosphere", NASA TN D-8445.

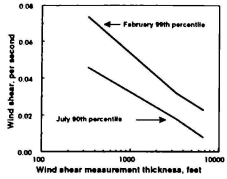


Figure 1. Jimsphere Vertical Wind Shear vs Measurement Thickness for Two Probabilities

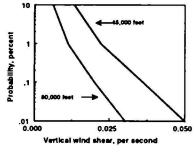


Figure 2. Example Maximum Vertical Wind Shear Probability from Rawinsonde Data.

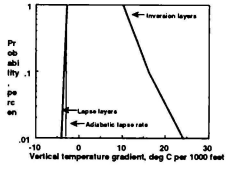


Figure 3. Maxumum Temperature Gradients Within Layer for Selected Probabilities.

LOW COST, REAL-TIME SIMULATION OF AN UNMANNED AERIAL VEHICLE

Michael E. Drews* **and Douglas A. Forman†**
Octant Technologies, Inc., Sunnyvale, California 94086

Abstract

The Vehicle Control and Simulation System (VCSS) provides the core unit of a portable, rapidly deployable ground control system for an Unmanned Aerial Vehicle (UAV). The system decommutates vehicle telemetry, and can compare the results in real-time with 6DOF mission simulations, running identical flight software as the UAV. While remotely commanding the vehicle, the pilot can visualize the mission by observing telemetered position and attitude of the vehicle in a 3D terrain following environment. Although the same unit can meet all of these mission requirements, it was developed under an extremely modest budget. This is made possible through a Rapid Prototyping approach to developing real-time simulations and utilization of Commercial Off The Shelf (COTS) hardware and software components to build the simulation system.

Background

A Vehicle Control and Simulation System (VCSS) is being developed to enhance ground control of an Unmanned Aerial Vehicle (UAV) through use of flight visualization and concurrent real-time mission simulation. The Phase 1 version of the VCSS was delivered to the Naval Air Warfare Center (NAWC) Advanced Technology Test Team (ATTT) at China Lake in November of 1998. The Phase 2 system is scheduled for delivery in August of 1999.

The primary goal of the VCSS R&D effort is to produce a system that will significantly enhance the current ground control system while keeping costs to a modest level. The VCSS has three core capabilities, and these are:

Flight Simulation:
Real-Time simulation of the mission using a six degree of freedom (6DOF) model of the UAV and a replicate copy of the flight software.

Vehicle Communication:
UAV telemetry decommutation, UAV joystick commanding, ground cockpit I/O interface to UAV RF system, and UAV telemetry simulation.

Data Distribution:
Display of UAV telemetry, display of concurrent mission simulation data, Heads-Up Display (HUD), 3D flight visualization, and telemetry distribution to other clients.

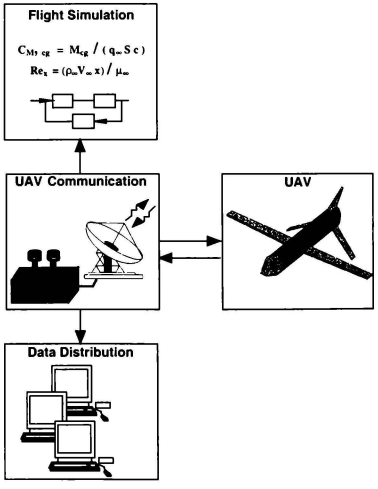


Figure 1 VCSS Core Capabilities

---

* Staff Controls Engineer, AIAA Member.
† Staff Software Engineer.
[1]Copyright © 1999 by Octant Technologies, Inc. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.



Figure 2 Telemetry Display Mode

To meet the system requirements, the VCSS was engineered to have several different operating modes, each one representing a combination of the previously described core capabilities. By mixing and matching the core modules, the operator can use the VCSS to perform a variety of different operations. One of the most basic operation modes, that of decommutating and displaying telemetry, is shown in a block diagram format in Figure 2.

Bi-directional communication is necessary for the operator to be able to interrupt UAV commanding by the way-point guidance flight software, and begin pilot-in-the-loop operations. The configuration that supports this feature is depicted in Figure 3. This configuration requires A/D and D/A boards in the VCSS to interpret joy stick and "cockpit" console commands, and communicate them to the UAV range RF equipment.

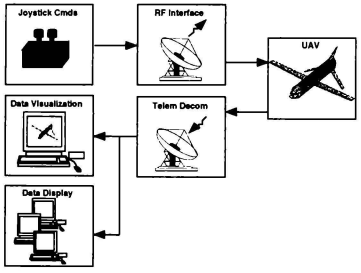


Figure 3 Pilot-In-The-Loop Mode

Additional operating modes are possible by including the 6DOF UAV simulation and flight software in the operational scenario. For example, although the primary purpose behind real-time concurrent mission simulation is to act as a pilot warning system, the same joy stick interface and flight models running on the VCSS allows the pilot to practice flying a simulated mission, even without the UAV in the loop. A block diagram description of this Pre-Flight Training Mode is displayed in Figure 4.

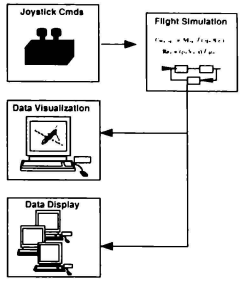


Figure 4 Pre-Flight Training Mode

System self-diagnostic capability is enabled through use of a telemetry simulation card and a configuration similar to that in Figure 4. The simulated mission (6DOF and flight software) is used to drive the VCSS telemetry simulation card, which can then be fed back into the VCSS telemetry decommutation cards, and used as a high fidelity, dynamic loop-back test of telemetry display, distribution, and visualization aspects of the VCSS. Figure 5 below depicts the system loop-back capability. Additionally, Table 1 on the following page summarizes how the three core capabilities of the VCSS can enable various operating modes.

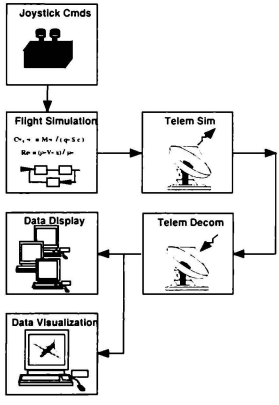


Figure 5 Dynamic System Loop-Back Capability

	FLIGHT SIM		VEH COMM				DATA DISTR		
SCENARIO	**6DOF Sim**	**Flight SW**	**Telem Decom**	**JoyStick Cmds**	**Telem Sim**	**Console Cmds**	**Telem Displ**	**Sim Data Displ**	**3D Visual**
Telemetry Display			x				x		
Pilot-In-The-Loop			x	x		x	x		x
Pre-Flight Training	x	x		x		x		x	x
System Loopback	x	x	x		x		x	x	

Table 1VCSS Matrix of Operational Scenarios

All of the hardware of the VCSS system, outside of PC and SGI visualization clients, is mounted into a portable and rapidly deployable 21 rack VME chassis. Once the system is set-up, the pilot can choose which of the operational scenarios outlined in Table 1 will be run by simply downloading a "scenario file" from a host PC to the VCSS embedded environment. The scenario file determines which portions of code will be run, the interconnectivity between the code modules, the configuration of the necessary I/O devices, and which client processes to invoke to display data. Use of the scenario operational paradigm is a direct result of the choice of tools used for the simulation software development. This choice highlights the development methodology itself, which was a key to delivering a highly capable system on a relatively low budget.

This methodology can best be summarized as follows: use of Commercial Off The Shelf (COTS) products whenever they meet project requirements, and utilization of Rapid Prototyping tools to develop mission unique software and hardware when COTS products fall short. The merit of using COTS products goes beyond the cost advantage of volume sales. COTS products typically undergo industry funded advancement, and support a host of widely used interfaces, which helps the vendor (and customer) maintain compatibility with the maximum number of external applications.

The benefit of using Rapid Prototyping software, such as Computer Aided Software Engineering (CASE) tools, is also twofold: CASE tools represent a more cost efficient method to realize mission unique code development, as well as allowing for rapid prototyping of baseline algorithm concepts. As these code streams are quickly brought to the real-time environment and tested, effective risk reduction is accomplished. Additionally, if multiple sub-system teams use the same CASE tools, integration is simplified, and the potential for code reuse increased.

The thrust of this paper is to exemplify how this methodology was used to engender the highly capable Phase 1 version of the VCSS, while simultaneously minimizing R&D and production costs.

Simulation Software Development

Instead of lauding the advantages of CASE tools further, this section will concentrate on how these tools were actually used under the Phase 1 effort.

Development of the VCSS occurred in conjunction with an avionics hardware and flight software upgrade for the UAV, which was managed by the NAWC Advanced Technology Test Team. This hardware and software package together is termed the Universal Replacement Auto-Pilot (URAP), and was built by Boeing's Phantom Works[1]. The URAP system is a reapplication of a design from the Joint Direct Attack Munition (JDAM) program, and was modified for the UAV to provide more realistic simulations of projected missile threats for ship self-defense[2].

The URAP team utilized the MATRIX$_X$™ CASE tool to develop the URAP algorithms, to build simulations to test these algorithms, and to eventually automatically generate real-time flight code ("AutoCode" ™) for the vehicle. The VCSS team at Octant also used MATRIX$_X$ for software development, and additionally, as the software integration environment for other

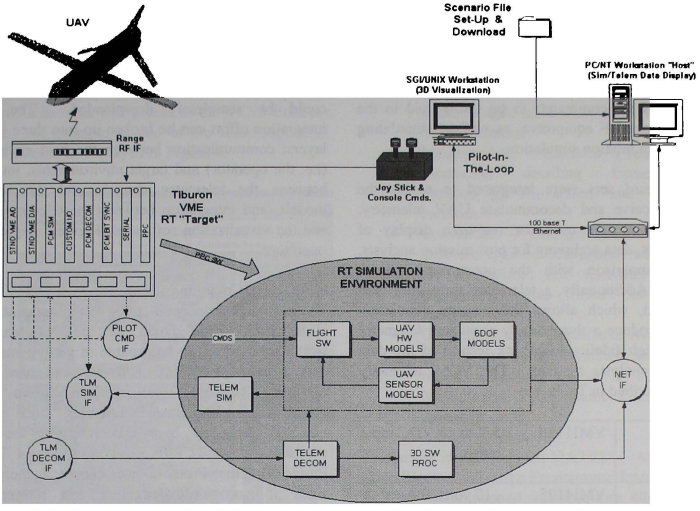


Figure 6 VCSS System Architecture and Simulation Environment

aspects of the VCSS code stream that were hand coded. The VCSS team discretized the continuous truth models used to test the flight software, and optimized the code stream for real-time simulation[3]. The discretized truth models were Autocoded in C, and with a replicate copy of the flight software, provided a real-time code stream of both the UAV and the mission environment.

The platform that was utilized to host the UAV simulation, as well as provide the vehicle/pilot interface, is also a COTS product, the Tiburón™ Simulation System. Tiburón is composed of hardware and software elements that aid in rapid prototyping and testing of real-time code streams. The "scenario" organization of the different VCSS modes described previously is a direct result of using the scenario building tools provided by Tiburón. Each scenario details which simulation or communication modules are to be download at run-time.

One drawback of using a CASE code generation tool is that when simulations grow in size, the resultant piece of generated code becomes difficult to manage. Compilation times also grow exponentially with code size. Tools that promote smaller, modular code size are beneficial because they aid the configuration management process (via unit testing) and improve the efficiency of the development cycle by minimizing compilation times.

The Tiburón system supports this approach by encouraging developers to create smaller, separate models, and then providing the post-processing glue that allows the separate automatically code generated models to be connected at run-time[4]. This integration is managed by the information that the user specifies in each scenario file.

Once the code is prepared for real-time execution, it is downloaded from the development system or "Host Environment" (typically an NT or Unix workstation) to the embedded system, or "Target Environment." In the case of the VCSS, the target environment is composed of a PowerPC processor card (PPC), running VxWorks™ Real-Time Operating System (RTOS) and mounted in a VME chassis. This is essentially the core hardware configuration of the Tiburón simulation platform. The VME chassis is outfitted with a host of other I/O cards to provide the required interfaces for

vehicle communication. The basic system architecture is outlined in Figure 6.

The pilot controls the UAV through both joystick and console types of inputs. To process the pilot commands, the card set detailed in Table 2 is used, which allows for commands to be transmitted to the UAV via range RF equipment, as well as stimulating the concurrent mission simulation.

Additional card sets were integrated to enable the VCSS to receive and decommutate UAV telemetry. Vehicle telemetry is used for real-time display of mission status, data archiving for post mission analysis, and for comparison with the concurrent mission simulation. Additionally, a telemetry simulation card was included, which allows the mission simulation models to produce a dynamically populated telemetry stream for high fidelity loop-back testing of the data display and analysis systems. The VCSS telemetry cards are described in Table 3.

Optically-Isolated Bi-Level Inputs	VMI1150	5 V to 48 VDC input range
Analog Outputs	VMI4105	±10 volts, 8 channels
Analog & Bi-Level Inputs / Outputs	VMI3114	8 Analog inputs, 2 Analog outputs, and dual 8-bit bi-directional digital input/output ports
Serial Dual Full-Duplex Inputs / Outputs	IP-MP Serial	Asynchronous serial communication to 10 Mb/s; EIA-232, EIA-422, and EIA-485 interfaces

Table 2 VCSS Phase 1 A/D & D/A I/O Capability

PCM Bit Synchronizer	SBS Berg 4400-VF-10	250 bps to 10 Mbps
PCM Decommutator	SBS Berg 4411-VX	100 bps to 6.0 Mbps
PCM Simulator	SBS Berg 4416-VF	1.0 bps to 10 Mbps

Table 3 VCSS Phase 1 Telemetry Capability

System Integration

With the requisite system components in place, the task to create an operational simulation system focused on component integration. Physical integration is quite trivial, as all the of the hardware VCSS components are COTS boards, constructed for operation in a VME backplane environment. The real challenge occurred in developing software interfaces which process all of the data streams such that each of the operational scenarios could be seamlessly downloaded. The overall integration effort can be broken up into three interface layers: communication between the host environment (i.e. the operator) and target environments, integration between the telemetry streams and the simulated models, and integration between the telemetry stream and the visualization software. Figure 7 depicts these interfaces.

Host/Target Communication:
The host/target concept dictates that systems not on the VME back plane (such as the NT and Unix workstations) need to be capable of getting and setting data from/to the VME software and hardware at runtime. The low level interface chosen for this purpose was 100 Base-T Ethernet, since standard transport protocols over this interface are widely available. TCP/IP (Transmission Control Protocol/Internet Protocol) was chosen as the protocol since it has acknowledgment and retransmission capability built into its fundamental core, yielding reliable transmissions.

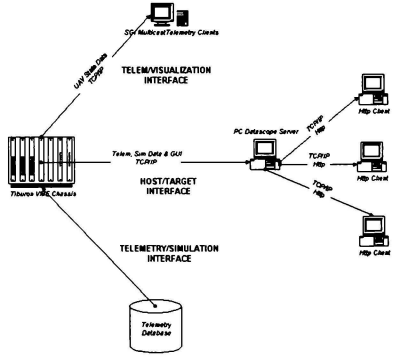


Figure 7 VCSS System Interfaces

For this layer, no new software had to be developed, because Tiburón provides two standard TCP/IP interfaces for interacting with the runtime code, an embedded HTTP Server, and a Stream Server. A Java based GUI (termed "DataScope" ™) uses the HTTP

Server to set inputs and the Stream Server is used to receive outputs, which it displays on a set of user definable screens. From the screens built in an editing mode, DataScope creates HTML pages at runtime. These panels are displayed with a standard web browser using the DataScope plug-in, and viewable by any other client with a browser and access to the VCSS Ethernet network. An example of one of the DataScope GUI operator screens is displayed in Figure 8.

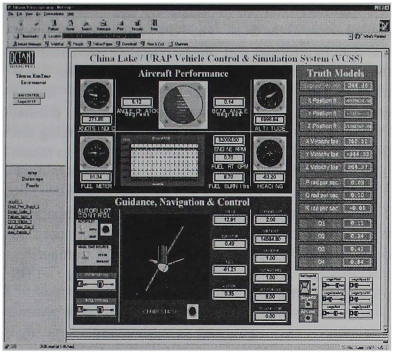


Figure 8 Simulation and Telemetry Data Display

Telemetry/Simulation Interface:
To be able to compare the mission simulation data with the real-time flight data, an interface layer between the telemetry decommutation cards and the simulation data had to be developed. To increase the possibility of code reuse, it was the intention of the VCSS team to create this interface in such a way as to be easily re-configurable to accommodate the different telemetry streams of other vehicles.

TelemBase™ was the database driven, "middleware" code module that was developed to perform the tasks of telemetry extraction for telemetry display purposes, and telemetry insertion, for the telemetry simulation capability. Telemetry extraction is the processing of raw telemetry frames in order to obtain scaled engineering units. Telemetry insertion is the processing of sets of engineering units in order to construct sub-commutated raw telemetry frames. TelemBase is termed middleware because it resides in the middle, between a raw telemetry device and an engineering unit value, or visa-versa.

Four tab-delimited files, which can be constructed with spreadsheet applications, completely define all of the information that TelemBase needs to decommutate the telemetry frame, and pass it on to the simulation:

- Frame.tlb – Contains information about the frame length, data rate, byte indexing and frame counters.
- Measurand.tlb – Contains a record for each measurand detailing frame location, transcoding, conversion and commutation methods.
- Convert.tlb – Contains a record for each different type of conversion method used to convert to/from raw telemetry counts and engineering units (used for polynomial scaling).
- Commutate.tlb – Contains a record for each type of commutation method, and determines which minor frames of a Time Division Multiplexed (TDM) frame a measurand will appear in.

TelemBase implements the $MATRIX_X$ /AutoCode User Code Block Application Programmer Interface (API) to feed the decommutated real-time data into the mission simulation for real-time comparison. The results of the comparison can be found through differencing like model/telemetry values, and sending the outputs to the operator via the TCP/IP host/target interface.

It is worth noting that the process of converting the flight software spreadsheet description of the telemetry stream into the TelemBase files was automated using the free software utility, PERL (Practical Extraction and Report Language). This automation eliminated hours of typing and minimized data entry errors.

Visualization:
With the host/target and telemetry/simulation interfaces completed, the last task was to give the UAV pilot the capability to better visualize the mission. Again, a COTS product was used, Paradigm/Multi-Gen™, which was integrated with the VCSS to generate 3D, terrain following images, coupled with HUD types of displays.

The Paradigm software is hosted on an SGI Octane workstation. The rationale for choosing Paradigm includes the following points:

1) Paradigm is a mature product/company, with a wide user base in virtual reality and military applications.

2) Multigen and Lynx allow the user to create complex scenarios in a GUI environment without writing/debugging any code.
3) The Paradigm Vega API provides complete access to underlying object methods that perform complex transformations and efficient 3-D graphical implementations. Because of the object-oriented approach of Paradigm, the user can manipulate objects created with Multigen and Lynx, by merely calling the object's member functions.

The CASE software package has these components:

- Multigen, a GUI based tool, which allows the user to build 3-D objects, including terrain maps.
- Lynx, a GUI based tool, which allows you to put together objects, simulate different dynamics and views, and save the configuration in a file.
- Vega, which is an API providing the user with functions that can read the Lynx file, create objects, modify objects, and apply rotational and translational dynamics to the objects.

To visualize the UAV position and attitude during real-time with the Paradigm/SGI system, an interface had to be developed to pass the telemetry data to the SGI. This was accomplished through the use of an API the VCSS team developed that allows reception of multicast data from the stream server. This module receives stream data (i.e. telemetry) from the VCSS target environment, and then multicasts it on a well-known group address and group port. Use of a multicast transmission format allows for easy expansion of network capability for a real-time critical network environment[5].

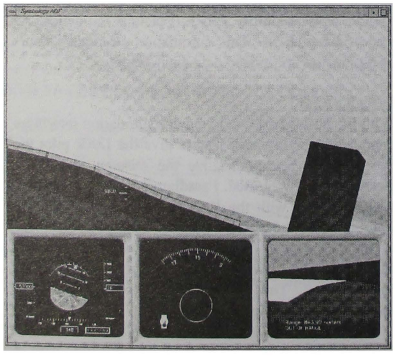


Figure 9 VCSS Real Time Flight Visualization

In the Phase 1 VCSS implementation, a proof of concept approach was taken to establish and verify the interfaces using "demo" terrain environments provided by Paradigm. The pre-built Paradigm Lynx terrain map was linked with code utilizing the Vega API library, to allow the telemetry information to stimulate the graphical environment. An example of the visualization capability provided by the VCSS is shown in Figure 9.

At run-time, the real-time UAV telemetered attitude and position data is multicast by the VCSS target environment to the SGI. The API on the SGI interprets this data in a format that the graphical environment can then use to render a "cockpit" view of the UAV maneuvering about the three dimensional terrain environment. With future development phases, Paradigm will be used to provide a view, looking out the nose of the UAV, of a terrain map created to match the terrain over which the UAV is actually flying.

The phased implementation of visualization capability is reflective of the risk reduction approach used to develop the VCSS. The following section summarizes this process and the current status of the VCSS prototyping effort.

Current Status

Phase 1 of the VCSS development centered on establishing that the key components of the system could be operated in tandem under the constraints of a real-time environment. A preview of the flight code, in addition to the full mission simulation, was translated to the real-time environment. The telemetry decommutation and simulation software and interfaces were also developed. Manual commanding was provided by a serial connection to a set of joysticks and GUI prototypes of the cockpit consoles were created. From the host PC, the pilot could utilize these GUI consoles in place of their hardware counterparts. The VCSS/SGI interface was also established, which allowed for flight visualization based on UAV telemetry data, utilizing a "demo" 3D terrain following environment.

The successful real-time factory demonstration of this system in November, 1998 paved the way for Phase 2 development, as well as spawning a copy of the Phase 1 prototype for field use. The thrust of Phase 2 is to enhance the complement of Phase 1 capabilities to begin Hardware-In-The-Loop (HWIL) testing of the VCSS and the UAV. These upgrades include a later release of the flight software, A/D and D/A interfaces to

allow hardware versions of the pilot console to be used, interfacing to RF equipment, and improved HUD screens on the SGI. Additionally, the network infrastructure is being developed to allow multiple clients to view and analyze real-time data. This Phase will culminate in ground HWIL testing in August 1999.

Follow on work will entail in-flight pilot-in-the-loop control and 3D terrain mapped visualization. Concurrent real-time mission simulation will provide a dynamic "expert system" capability that can not only aid a pilot flying a singular UAV, but eventually assist in the monitoring and control of several unmanned vehicles in formation flight. The modularity of the simulation environment provides an easily expandable platform for simulation of multi-vehicle missions. Multiple vehicles can be managed via the scenario file concept, and additional COTS processors can be integrated into the VCSS to support the increased computational demands.

Conclusion

Throughout the last decade, the rapid performance increase in commercial computer processors has given engineers tremendous resources to simulate complicated, dynamic systems. At the same time, the cost of such computational power has dropped, making highly capable simulation systems more available, and instituting a relationship that continues to drive performances higher and prices lower. This phenomenon is not isolated to computer hardware, but includes other key components of simulation development: mature rapid prototyping tools, stable code generation techniques, and widely available sets of A/D, D/A and telemetry processing cards. Equally as important is the momentum towards open communication architectures and languages.

The development methodology of the VCSS system utilized the advances and portability of commercial technologies to rapidly synthesize a real-time simulation system. By minimizing unique code development and eliminating unique hardware development, the Phase 1 system was put together in a low level funding environment ($250K*), and in a relatively short period (5-person-months, spread over nine calendar months).

There are obvious drawbacks to reliance on commercial systems. Ultimately, the user is reliant on vendor commitment to backward compatibility, equipment availability, and technical support. Additionally, code and hardware that can support many uses can have penalties in efficiency or resource utilization (power, volume, etc) when compared with custom equipment.

However, utilizing CASE tools and COTS products when possible, enables rapid development cycles and cost savings as well. The coupled effect of these benefits is that the time and financial resources needed for prototyping is reduced, consequently reducing development risk and creating pathfinder opportunities. Such a pathfinder opportunity defined the goal of the VCSS project: to utilize methodologies and technologies that allow for rapid and low cost development of a UAV control and simulation system. The current operational capability and cost of the VCSS has validated this approach for rapid system development projects.

Acknowledgements

The authors gratefully thank the Naval Air Warfare Center Advanced Technology Test Team at China Lake, and the URAP team at Boeing/Phantom Works, for their invaluable contributions towards development of the VCSS.

References

[1]Bruns, J., *Functional Design Document for the Replacement Auto-Pilot (RAP) System*, Boeing Phantom Works Advanced Tactical Missiles, St. Louis, Missouri, 1998.

[2]N.A.W.C. Advanced Technology Test Team, "New Autopilot Successfully Completes First Phase of Flight Test", *Press Release*, China Lake NAWC, 1998.

[3]Octant Technologies, Inc., *VCSS Phase 1 System Capabilities Document*, OTI Report Number 6700-6833-002, Sunnyvale, California, 1999.

*Cost includes the following: Software tools (Code Generation and Visualization), Hardware (real-time target processors, A/D, D/A boards and Telemetry Cards) and Labor. NT and SGI workstations not included.

[4]Maufer, T. *Deploying IP Multicast in the Enterprise*, Prentice Hall, New Jersey, 1998.

[5]Octant Technologies, Inc., *Tiburón User's Manual*, Sunnyvale, California, 1999.

RANDOM CHOICE MODEL FOR THRUST PROFILE

Waldemar C. Leite Filho† and Pelson Souza Pinto‡
Instituto de Aeronáutica e Espaço,
12228-904 - S. José dos Campos (SP) Brazil

ABSTRACT

In this work the problem of random choice for thrust profiles, during the simulation of trajectories using Monte-Carlo's method, is discussed. The data of each motor are supplied in the form of three curves, so called nominal, superior and inferior. Starting from these curves, it is defined limits, inside of the which the real thrust profile should be. The problem becomes, then, to find a way to generate, during the execution of simulations, random profiles of the thrust within the supplied limits, just using one random choice. The detailed simulation scheme is presented as well as the simulation results.

INTRODUCTION

The complexity of aerospace vehicle control systems and the high cost of even simple flight tests lead the application of simulation in aerospace vehicle control design to be extensive and absolutely necessary.

In this work the problem of random choice for thrust profiles, during the simulation of trajectories using Monte-Carlo's method, is discussed. Monte-Carlo's studies are used widely so much for specification as for verification of acting of systems, due, mainly, its easiness of use combined with good results. Although the method of treating random variables is classic, we found some difficulty in the treatment of some types of random processes, in particular in those where the event time duration is also random. The work here described proposes a solution for this difficulty, in particular in what it refers to the thrust and mass profiles.

During a Monte-Carlo study, to each new simulation a thrust profile is chosen, that should be between the inferior and superior limits defined and it is of great convenience that this choice can be done with only one random variable and such that this profile generated has the same properties of those used as limits.

The thrust profiles (and its corresponding mass profiles) are obtained from theoretical calculations and also from results of experiments. The data of each motor are supplied in the form of three curves, so called nominal, superior and inferior (see Fig. 1). Starting from these curves, limits are defined, inside of the which the real thrust profile should be placed. The problem becomes, then, to find a way to generate, during the execution of simulations, random profiles of the thrust within the supplied limits, just using one random choice.

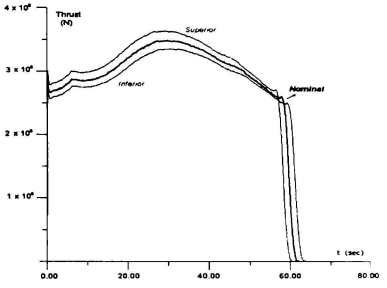


Figure 1 – Limits of Thrust Profile

METHOD DESCRIPTION

The difference of the burning time between the nominal profile and the dispersive case is the main cause of difficulty. Given a rocket motor, its specific impulse is determined. So the smaller time it burns, the larger value of the thrust appears which is not simply proportional to the nominal profile. Vice-versa it happens for a time of it burns larger than the nominal. Due to this it is not possible to assign a new profile just multiplying the nominal one by a gain (changing only its amplitude). Moreover, if the difference of the burning time is not considered, the assignment a new profile by a gain implies in changing of the total energy.

The problem was split in two parts: firstly it was defined which properties an operator (over the nominal profile) should have to produce a new profile between

† Senior Researcher of Space System Dept., AIAA Associate Member

‡ Senior Researcher of Space System Dept.

Copyright©1999 by American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

the limits, through a random choice; after it was built an operator which has these caractheristics.

The problem can be set as following:

given $f_n(.), [0, T] \rightarrow \Re$ nominal profile

and $f_d(.), [0, T] \rightarrow \Re$ dispersive profile

to find a random profile $f_a(t), [0, T] \rightarrow \Re$ between $f_n(t)$ and $f_d(t)$, such that

$$\int_0^T |f_n(t) - f_d(t)|dt \geq \int_0^T |f_n(t) - f_a(t)|dt . \quad (1)$$

The constraint (1) was defined in the integral form because the simple absolute value form would lead

$f_a(t) = f_n(t)$ when $f_n(t) = f_d(t)$,

and such constraint is not necessary.

The operator ***P*** should has the following properties:

$\boldsymbol{P}[f_n(t),0] = f_n(t)$

$\boldsymbol{P}[f_n(t),1] = f_d(t)$

$\boldsymbol{P}[f_n(t),\alpha] = f_a(t), \ \alpha \in (0,1)$

To get a compromise between simplicity and accuracy, it was proposed the following operator ***P***:

$$P[f_n(t), \alpha] = (1 + d_1.\alpha).f_n[(1 + d_2.\alpha)t] \quad (2)$$

where $f_n(.)$ is the nominal thrust profile, d_1 and d_2 are adjustable coefficients and α is a random parameter. The coefficients d_1 and d_2 are selected for each pair of nominal and limit (superior and inferior) profiles. The parameter α defines the statistics of the generated curves set, in such way that with only one random variable is enough to define a new curve. The parameter α is a random variable that presents the same probability distribution of the thrust profile process. Figure 2 shows the probability distribution used – near Gaussian.

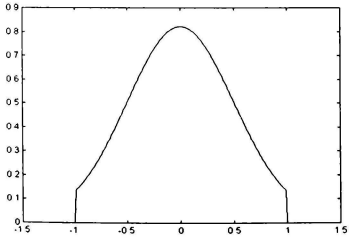


Figure 2 – Probability Distribution of α

Parameters d_1 and d_2 are obtained from the criterion of minimum quadratic error. As the new profile can be above or below of the nominal case, it is necessary to consider two sets of d_1 and d_2. When $\alpha > 0$ it is considered d_{1Sup} and d_{2Sup} such that

$$\{d_{1Sup}, d_{2Sup}\} = \arg \min_{d1,d2} \int_0^T [(1+d_1).f_n((1+d_2).t) - f_{sup}(t)]^2 dt$$

When $\alpha > 0$ it is considered d_{1Inf} and d_{2Inf} such that

$$\{d_{1Inf}, d_{2Inf}\} = \arg \min_{d1,d2} \int_0^T [(1+d_1).f_n((1+d_2).t) - f_{Inf}(t)]^2 dt$$

Once d_1 and d_2 determined, the simulation can be done calculating the thrust profile by eq. (2) for each new case. Figure 3 shows schematically this process.

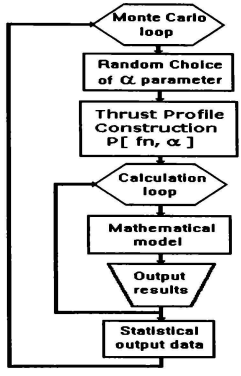


Figure 3 – Monte Carlo Block Diagram

RESULTS

It follows the values found to first stage of VLS[1].

$d_{1sup} = .0484$; $d_{1inf} = -.0332$

$d_{2sup} = .0250$; $d_{2inf} = -.0244$

Using those values for d_1 and d_2 it is possible to generate random of thrust profiles by eq.(2). Figure 4 shows the whole set of such profiles.

COMMENTS AND CONCLUSION

The methodology here proposed came to solve the difficulty found in the generation of random thrust profiles. Its implementation is easy and its execution is fast. This improves the overall performance of Monte Carlo simulations, allowing their utilization during the

trajectory studies of the Brazilian Satellite Launcher. Besides, it is sufficiently general to be applied in similar cases where one wants to generate curves inside of specified limits.

REFERENCES

[1] Leite Filho, W.C. & Carrijo, D.S., "Hardware in The Loop Simulation of Brazilian Launcher VLS", *Proceedings of 3rd ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*, Noordwijk, The Netherlands, Nov. 1996, pp. 355-358.

[2] Yoshino, T., & Boscov, J., "Main Propulsion System of Brazilian Satellite Launch Vehicle VLS", *Proc. 16th Inter. Symp on Space Technology and Science*, Japan, 1988, pp. 149-155.

[3] Malyshev, V.V. et alii, "Launcher Guidance Algorithm Design", *Aerospace Vehicle Control: Modern Theory and Applications*, IAE, S. José dos Campos, Brazil, 1996, pp. 303-344.

[4] Carrijo, D.S., Oliva, A.P. & Leite Filho, W.C., "Hybrid Simulation Software Scheme for Validation of a Launcher Control System", *Preprints of 14th IFAC Symposium on Automatic Control in Aerospace*, Seoul, Korea, Aug. 1998, pp. 427-432

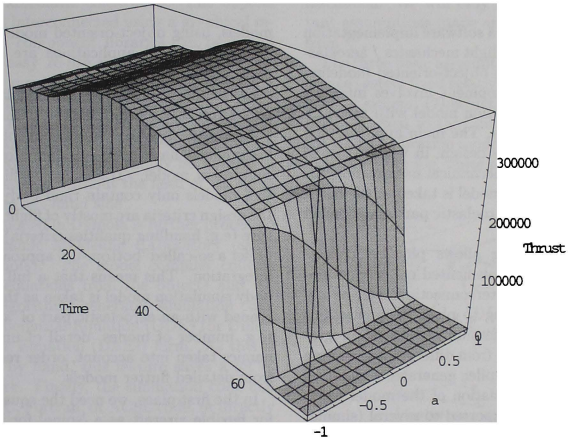


Figure 4 – Complete Set of Thrust Profiles

Integrated Flight Mechanics and Aeroelastic Aircraft Modeling using Object-Oriented Modeling Techniques

Gertjan Looye*

German Aerospace Center
DLR-Oberpfaffenhofen
Institute for Robotics and System Dynamics
D-82234 Wessling, Germany
E-mail: gertjan.looye@dlr.de

Abstract In this paper a software implementation is proposed of integrated flight mechanics / aeroelastic aircraft models, using object-oriented modeling techniques. Model development involves integrating a rigid aircraft simulation model with aeroelastic flutter analysis models. The main application is primary flight control law design, in which most of the criteria are of flight mechanical nature. For this reason, the rigid aircraft model is taken as the basis, while the fidelity of the aeroelastic part may depend on the accuracy required.
Object-oriented modeling allows physical objects and phenomena to be implemented one-to-one into software objects, since interconnections can be defined freely (e.g. according to physical interactions like energy flow). This feature facilitates integration of model components from different engineering disciplines. A model compiler generates simulation code by symbolic manipulation of the model equations. The code can be exported to several (simulation) languages. This allows the same model to be used in different engineering environments. We illustrate this feature with a flutter analysis example.

1. Introduction

This paper proposes a software implementation of integrated flight mechanics / aeroelastic aircraft models, using object-oriented modeling techniques. The main model applications are simulation and flight control law design.

For slender airframe designs, like large civil transport aircraft, structural flexibility may considerably influence the flight dynamics. For this reason, aeroelastic effects have to be accounted for in the aircraft simulation model. Traditionally, flight control design models only contain rigid body dynamics and the design criteria are mostly of flight mechanical nature (e.g. handling qualities criteria). Therefore, we prefer a so-called 'bottom-up' approach in the model integration. This means that a full nonlinear rigid body simulation model is taken as the basis, and extended with an aeroelastic part of 'scalable' fidelity (e.g. number of modes, detail of unsteady aerodynamics taken into account, order reduced), derived from detailed flutter models.

In the first place, we need the equations of motion for flexible aircraft as a 'spine' for interconnecting the different model components. These equations are derived in refs [12, 20, 4].

Given the equations of motion, we can start thinking how to interconnect models that actually 'drive' these equations, like the aerodynamic model and thrust models. In this paper we will concentrate on implementation and interconnection of these submodels, rather than the modeling itself.

The total aircraft model consists of many submodels from different engineering disciplines. This is

*Research Engineer

Copyright ©1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

an important reason why we use the object-oriented modeling language Dymola (DYnamic MOdeling LAnguage[1]) [5]. Moormann [14, 13] successfully implemented a rigid aircraft library using this language. This paper describes the further development to flexible aircraft. The Dymola language allows one-to-one implementation of physical objects and phenomena into software objects, because of freedom in defining the object interconnections. These interconnections are not limited to signal flows (like in most control-oriented graphical simulation tools) but represent physical system interactions, like energy flows, or kinematic constraints. The objects are defined and interconnected using a graphical interface. Model component libraries can be developed to allow easy re-use in other models [14, 13]. These libraries may contain different objects for a same sub-model with different levels of detail. These objects can be easily exchanged within the aircraft model. The modeling language is equation based, i.e. the model components can simply be written in the form of equations, without the need of 'bringing unknowns to the left-hand side'.

A model developed in Dymola is further processed using the DYnamic MOdeling LAboratory (same acronym: Dymola). The model compiler collects and sorts the equations from the objects and using symbolic algorithms generates simulation code in Ordinary Differential Equation (ODE) (or Differential Algebraic Equation (DAE)) form. As this is normally done by hand, this feature saves a lot of error-prone work. Finally, the simulation code can be exported as C, Fortran, or ACSL code, or directly to environments such as MATLAB/Simulink [11] and Matrix$_x$/SystemBuild [7]. This gives great flexibility in application of the model.

This paper is structured as follows. In Section 2 we review the equations of motion for flexible aircraft. In section 3 we briefly discuss aerodynamic model integration. Next we describe the implementation of an example model in Dymola. In section 5 we discuss code generation. In section 6 we present flutter analysis as an example application of the model. We will end with conclusions in section 7.

[1] to be updated to the Modelica standard in near future [6]

2. Equations of motion

Flight mechanics models are based on the nonlinear Newton-Euler equations of motion for a rigid body [3]. These equations describe the motion of a rigid aircraft with six-degrees of freedom, driven by thrust, aerodynamic and gravity forces. In the references [12, 20, 4] the equations of motion are extended to flexible aircraft. Ref. [4] presents a detailed derivation and shows how structural and rigid dynamics can be integrated in a mechanically correct way. The reader is referred to these references for a detailed discussion. We will only mention the most important assumptions made and cite the final equations from [20].

The three basic assumptions we make are (1) a reference system fixed relative to the earth surface is an inertial system; (2) structural deformations are sufficiently small so that linear elastic theory applies; (3) a set of vibration modes with eigen frequencies is available from e.g. finite element analysis. Figure 1

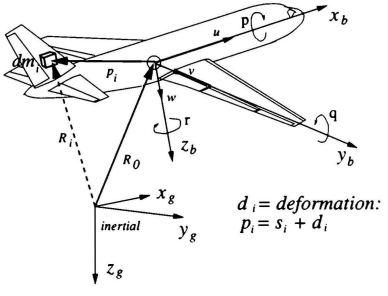


Figure 1: Flexible Aircraft

shows a free flying aircraft considered as a large set of lumped-masses dm_i (assumption 4), of which one is shown in the figure. The axes x_g, y_g, z_g form the inertial reference frame. The axes x_b, y_b, z_b form the body axes that move with the airframe. The inertial position of the origin is R_0. The angular rates of the body axes are $\Omega = [p,\ q,\ r]^T$, while the Eu-

ler angles are $E = [\phi,\ \theta,\ \psi]^T$ (not depicted). The components of the inertial velocity vector along the body axes are $V = [u,\ v,\ w]^T$. The position of the mass element dm_i is p_i, consisting of its undeformed location s_i and its deformed translation d_i. The local deformation is written as a linear combination of mode shape deformation vectors Φ_{ij}, where j refers to the jth elastic mode:

$$d_i = \sum_{j=1}^{N_e} \Phi_{ij}\eta_j \tag{1}$$

N_e is the number of modes taken into account and η_j is the generalized displacement of mode j. In refs[20, 4] the equations of motion are derived using Lagrangian mechanics. The most important issue is the choice of an appropriate body reference system. In all references, so-called *mean axes* are considered as the best choice, since this results in minimal inertial coupling between rigid and elastic dynamics. The orientation of mean axes is such that deformation induced translational and rotational momentum are always zero. In reality, these constraints are hard to satisfy, but with the assumption (3) that the deformations are small, and that deformation and deformation rates are co-linear, (assumption 5) it is sufficient when so-called *practical* mean axes constraints [20] are satisfied. If the mode shapes have been obtained from free-free finite element analysis, then a reference system with the axes in the direction of the rigid modes, and with the origin always in the center of gravity, satisfies these practical mean axes constraints.

Finally, we make the assumption that the inertia tensor is constant, although it will vary slightly due to deformation of the airframe (assumption 6). The resulting equations of motion are as follows:

* Kinematic relation R_0 and V:

$$\dot{R}_0 = R_{bg}^T(E)\ V \tag{2}$$

where R_{bg} is the rotation matrix from the inertial to the body mean axes.

* Kinematic relation E and Ω:

$$\dot{E} = R_{\phi b}(E)\ \Omega \tag{3}$$

where $R_{\phi b}$ is the transformation matrix from body angular rates into Euler angular rates.

* Rigid body force equation:

$$m[\dot{V} + \Omega \times V - R_{bg}(E)\ [0,\ 0,\ g]^T] = F_A + F_T \tag{4}$$

where F_A and F_T are aerodynamic and thrust force vectors respectively, and g is the gravity acceleration.

* Rigid body moment equation:

$$I\dot{\Omega} + \Omega \times I\Omega = M_A + M_T \tag{5}$$

where M_A and M_T are aerodynamic and thrust moment vectors respectively.

* Elastic degrees of freedom: $(j = 1...N_e)$

$$M_j(\ddot{\eta}_j + 2\zeta_j\omega_j\dot{\eta}_j + \omega_j^2\eta_j) = Q_{\eta_j} \tag{6}$$

where M_j is the generalized mass matrix, ζ_j the structural damping, ω_j the eigenfrequency, and Q_{η_j} the generalized aerodynamic force for mode j respectively.

The only coupling between rigid and flexible motions is via the aerodynamic forces and moments. Inertial coupling disappears through the assumptions made (5,6). Dropping these assumptions, the equations become considerably more complicated [4].

For flexible aircraft, the kinematics of local points on the airframe with respect to the mean axes are more complicated than for the rigid case. We need to know these local kinematics at the positions where we want to interconnect for example engine and sensor models. As we see from fig. 1, the kinematics of a local point are described by the movements of the mean axes (R_0), the undeformed location w.r.t. the mean axes (s_i), and the local airframe deformation (d_i, from eq. 1).

For our implementation (section 4) we will make use of multi-body principles. At the airframe locations where we intend to attach e.g. engine or sensor models, we define a local reference system which, contrary to the mean axes, is physically attached to the airframe. In undeformed position the orientation of the local and the mean axes are the same. Along the local body axes, we compute inertial kinematics of the local point (position, orientation, (angular)

velocity, (angular) acceleration), and forces and moments. For example, the relation between absolute and relative velocities of the local and mean axes are given by:

$$\dot{d}_i = R_{bl} v_{i_l} - V + p_i \times \Omega \qquad (7)$$

where the velocity of point i relative to the mean axes is:

$$\dot{d}_i = \sum_{j=1}^{N_e} \Phi_{ij} \dot{\eta}_j$$

v_{i_l} is the inertial velocity of point i along the local axes (l), R_{bl} is the time dependent rotation matrix from the local into the mean axes. This matrix depends on the local angular displacement vector ϕ_i. We will not cite the equations for acceleration here, but it will be clear that differentiating eq. 7 leads to a complicated expression [15].

An engine or sensor model now, is defined in its own reference system. Connecting a model to the airframe means that its local axis system and the local axis system defined on the airframe merge, i.e. kinematics of both axis systems are constraint to be equal. A vertical acceleration sensor for example, is modeled such that it outputs the acceleration along its vertical z-axis. Once interconnected with the airframe model, this acceleration is equal to the inertial acceleration in z-direction of the locally defined reference system on the airframe. As we will show in section 4, we can directly implement this interconnection procedure, without need for reordering equations or variables to sort out unknown variables; Dymola will do this automatically when generating simulation code.

3. Aerodynamic model

Although not exercised for the model example presented in this paper (since data was readily available), a major integration effort in developing an integrated flight mechanics/aeroelastic aircraft model is required in development of the aerodynamic model. Therefore we will briefly discuss some key aspects of the aerodynamic modeling.

Both for flight mechanic and aeroelastic models 'agreed-upon' aerodynamic modeling methods exist. In rigid models so-called stability and control derivatives are used, usually implemented in polynomial or tabular form. These derivatives are obtained from for example handbook methods, CFD computations, wind tunnel experiments, and flight testing. If necessary, flexibility of the airframe is accounted for using so-called rigid/flex static correction ratios that scale the derivatives.

The computation of aeroelastic forces and moments is highly complicated. A trade-off is necessary between accuracy and computational costs. For this reason the Doublet Lattice method[1] is a popular method in industry to compute aeroelastic models in sub-sonic flow regimes for flutter analysis. For our proposed model integration, we use those flutter models to develop the aeroelastic part of the aerodynamic model.

Integration of rigid and the aeroelastic aerodynamic models is for example discussed in refs [2, 22, 9]. The model consists of four parts:

$$\begin{bmatrix} Q_R \\ Q_\eta \end{bmatrix} = \begin{bmatrix} \mathrm{RR}(\underline{x}_a, \dot{\underline{x}}_a, ...) & + & \mathrm{RF}(\eta, \dot{\eta}, ...) \\ \mathrm{FR}(\underline{x}_a, \dot{\underline{x}}_a, ...) & + & \mathrm{FF}(\eta, \dot{\eta}, ...) \end{bmatrix} \qquad (8)$$

where $Q_R = [F_A,\ M_A]^T$ are forces and moments driving rigid dynamics, $\underline{x}_a$ denotes rigid body aerodynamic states, such as airspeed V_A, angle of attack α and sideslip angle β. η is the vector with generalized displacements of the elastic modes. Of course the model depends on other variables as well, such as control inputs. RR (Rigid-Rigid) and RF (Rigid-Flex) describe the forces and moments driving the rigid part of the equations of motion, induced by rigid and flexible aircraft states respectively. FR (Flex-Rigid) and FF (Flex-Flex) describe the generalized forces driving the flexible equations of motion, induced by rigid and flexible aircraft states respectively.

The RR part is the same as in the rigid aircraft model, whereas the three other components are to be obtained from flutter models. It is generally known that the computation of generalized forces induced by rigid modes in flutter models is poor, but up to now we did not look into other modeling methods.

Flutter models are obtained in the frequency domain and therefore need to be transformed into the

time domain. Several approaches have been developed that are based on rational function approximations in the Laplace domain[16, 8]. Ref. [17] compares several methods and concludes that Karpel's method[8] gives the most accurate results with the lowest number of additional aerodynamic lag states.

Rigid aerodynamic models implemented in look-up tables are usually valid over (a part of) the flight envelope, whereas flutter models only have very local validity. In practice, these models are computed over a grid of flight conditions and aircraft configurations. It is very difficult to parametrize these models as a function of such variables. Therefore, the most obvious way to increase validity of the aeroelastic parts in the aircraft model is interpolation between a set of locally derived aeroelastic models.

Finally, both the rigid and the aeroelastic parts of the aerodynamic model contain static aeroelastic information. In the rigid model this is represented by the rigid-flex ratios. In order that the trim state vector does not depend on the number of modes that is included, the refs. [22, 9] use a procedure developed by Adams at NASA Langley, in which the static influence is subtracted from the RF part of the aerodynamic model. Static aeroelastic deformation is represented by the rigid-flex ratios only.

4. Model implementation

Figure 2 shows the basic implementation of an example flexible aircraft model in the Dymola graphical interface. The aircraft model has been taken from ref. [21]. It has one asymmetrical mode and four symmetrical modes (see also ref.[20]). The quasi-steady aerodynamic model was derived using strip-theory.

The central object *flexbody* contains the equations of motion as discussed in section 2, and since all other objects are connected to it, it has indeed the function of a spine for the model. In addition two engine models are visible (the aircraft has four engines; each model actually represents a pair of engines on each wing), as well the aerodynamic, gravity, atmospheric, wind and systems (actuators for aerodynamic control surfaces) sub-models. In each

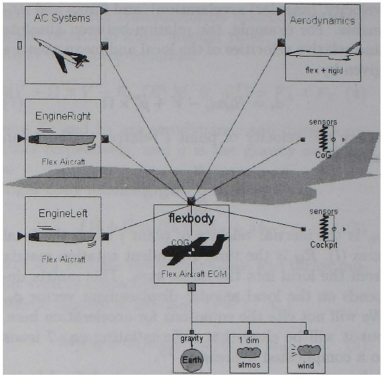


Figure 2: Model implementation (top level)

object a local reference system is defined in which the sub-model is described.

All objects have connection points to other objects. These connection points will be referred to as 'cuts'. Connections between cuts represent physical relations between sub-models, in this case kinematic constraints and energy flows. In these cuts the following variables are defined:

Forces and moments - F_l, M_l (force and moment vector along local reference system of the sub-model), Q_η (generalized aeroelastic forces).

Kinematic quantities - R_{gl} (rotation matrix from local into inertial axes), R_0 (inertial position vector of local axes origin), V, Ω (translational and rotational inertial velocities of the local axis system, described in local axes), a, α (translational and rotational inertial accelerations of the local axes), η, $\dot{\eta}$, $\ddot{\eta}$ (mode shape generalized coordinates and derivatives). Also environmental quantities are defined, but we will not discuss these. When connecting two cuts, the kinematic quantities in both cuts are set equal, so that the objects are forced to move exactly the same way. The forces and moments on the other

hand, are summed to zero (can be interpreted as a local equilibrium condition).

Flexbody object

The *flexbody* object has two cuts: the variables in the lower cut are in earth fixed axes, in the upper one in body mean axes. Wind, atmosphere and gravity are modeled in earth fixed axes, and are therefore interconnected via the lower cut. The other objects move with the aircraft and drive the equations of motion via the mean axes. These are therefore interconnected via the upper cut. The kinematic relations between earth fixed and mean axes are part of the equations of motion, see section 2.

Figure 3 zooms in on the *flexbody* object (by right-clicking on it with the mouse in the graphical interface). The lower right block (*eqm*) contains the rigid

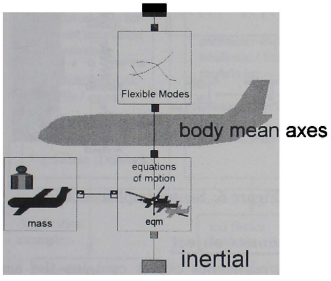


Figure 3: Equations of motion (EQM) object

equations of motion, entered in exactly the same form as in eqs. 2, 3, 4, 5. Again, the variables in the lower cut are in earth axes (inertial), whereas those in the upper cut are in body axes (6 DOF). The object to the left (*mass*) defines and computes mass dependent variables (empty aircraft weight, fuel weight, inertia tensor). The top-right object (*Flexible Modes*) contains the flexible equations of motion (eq. 6) in mean axes. By interconnecting the *eqm* and *Flexible Modes* objects, the mean axes merge with the body axes in the first object, i.e. the mean axes become a 6 DOF reference system. The variables in the upper cut of the second object are also in mean axes, but herein also generalized mode shape coordinates and derivatives (η, $\dot{\eta}$, $\ddot{\eta}$) are defined, as well as generalized aeroelastic forces (Q_η).

All objects are of generic nature, and can be used in any aircraft model. When double-clicking on an object, a parameter window shows up that allows the user to enter aircraft-specific parameters. The window for the *Flexible Modes* object is shown in figure 4.

The number of longitudinal and lateral modes can be adapted via *nmodeslon* and *nmodeslat* respectively, as a means of scaling the fidelity of the elastic part of the model. The eigenfrequencies can be adapted as well (*Wlon1, Wlon2*, etc.). The parameter *doResid* specifies what to do with the modes to be removed. If *doResid* is true, $\dot{\eta}_j$ and $\ddot{\eta}_j$ in equation 6 are set to zero so that remains:

$$M_j(\omega_j^2\eta_j) = Q_{\eta_j}$$

which will be solved by the model compiler. In the other case, η_j, $\dot{\eta}_j$ and $\ddot{\eta}_j$ are set to zero. The model compiler then automatically removes the equation in the simulation code, thus truncating mode j.

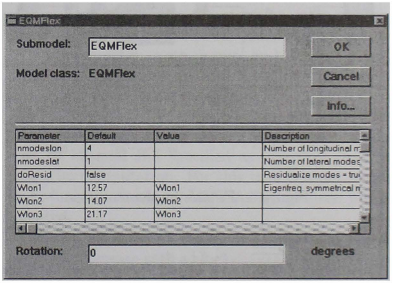


Figure 4: Parameter dialog window *Flexible Modes*

Engine model object

Figure 5 shows the contents of an *Engine Model* object. The *Thrust* block contains a highly simplified engine model that only scales a throttle input

into a thrust force (without dynamics). This thrust force is simply defined as $T = [T_x, 0, 0]$, where T_x is the thrust along the local x-axis. As discussed in the previous section, a local axis system (attached to the airframe) is defined at the engine location. This is done via four transformation objects, *TranslateFix, TranslateFlex, RotateFlex* and *RotateFix*. These relate the mean axes with the local axis system on the airframe via a fixed translation (from CoG to engine in undeformed position), a time dependent translation and rotation due to local airframe deformation, and a fixed rotation (engines are usually installed with a small angular offset) respectively. By connecting the engine model with the transformation objects, directional thrust variations with respect to the mean axes due to flexibility are automatically accounted for. The transformation objects are generic; they are used within the *sensors* object as well. Local airframe information (e.g. position, mode shape vectors) is specified via a parameter window.

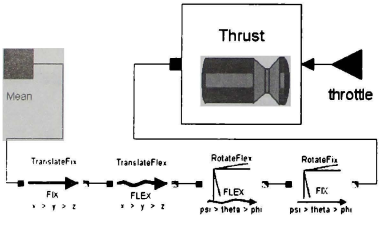


Figure 5: Engine implementation

Sensors object

In figure 2 two sensor objects are visible, *CoG* and *Cockpit*. Both are based on the generic object *sensors*, depicted in figure 6. On the right we see five (simple) sensor models that are modeled in their own axis systems. Like in the object *Engine model*, each local axis system is related to the mean axes via four transformations. The transformation objects are exactly the same as for the engines, only local position and mode shape information for the cockpit has been entered via the parameter window.

The transformation objects are based on a same parent object. This parent has the two cuts and implements general kinematic relations between the cut variables (e.g. eq. 7). The four transformation objects inherit these features and implement the actual kind of transformation (e.g. flexible translation).

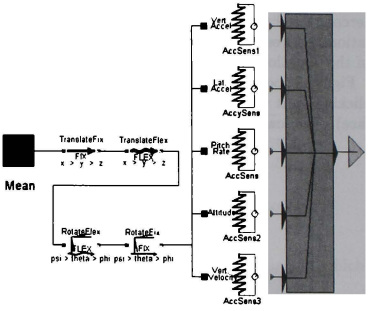


Figure 6: Sensor implementation

Aerodynamics object

The *Aerodynamics* object contains the aerodynamic model in the form of equation 8. The RR part is implemented in the form of stability and control derivatives that are interpolated from look-up tables. The RF, FR, and FF parts are implemented as constant matrices, valid for a single operating condition and aircraft configuration. The aerodynamic model is defined in mean axes.

5. Model processing

The model building process in Dymola is summarized in figure 7. At the top is the actual model implementation, as described in the previous section. A model is composed in the graphical user interface (GUI) from objects that are stored in libraries.

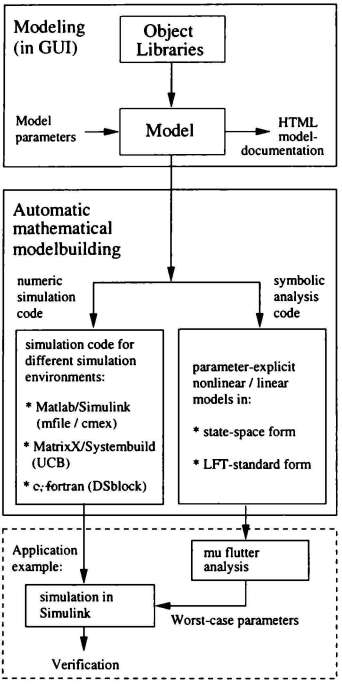


Figure 7: Model building process

Parameters can be adjusted to aircraft-specific values. An HTML-code generation facility is available for model documentation, see for an example ref. [13]. Once finished, the model is loaded in the Dymola main program. The symbolic equation handler collects all equations from the model objects and solves them according to the inputs and outputs of the aircraft model (*Automatic mathematical model building* figure 7). Equations that are not necessary in the simulation code are automatically removed. Symbolic simulation code is generated in nonlinear state space form. This code can be exported in several languages, like c, Fortran, but also for specific environments like MATLAB/Simulink (m-code, or Simstruct c-code) and Matrix$_x$/SystemBuild (User Code Block). Another possibility is to generate symbolic analysis code in state-space from, which can be used to symbolically generate a model in the form of a Linear Fractional Transformation using symbolic mathematical software, see ref. [19, 18].

Dymola offers a built-in trimming/simulation and visualization facility for direct model evaluation. However, for application such as control law design, the export facility to for example MATLAB/Simulink is extremely useful. We used Simstruct code to simulate the aircraft in this environment. With this code also an m-file is generated in which parameter, state, input, and outputs names and initial values are defined. This information can for example be used to automatically generate a GUI in MATLAB for trimming and linearizing the model, see figure 8. Within this GUI, trim values can be entered and specified as 'fixed' or 'free' in the trim computation. For this trim computation a mex-function based on MINPACK has been implemented.

Finally, the aircraft model can be included into a simulink diagram with a block diagram of the control system to perform closed loop simulations.

6. Application example

As an application example, we use the model for flutter analysis in the μ-framework [10]. The aircraft dynamics are augmented by a pitch rate feedback via

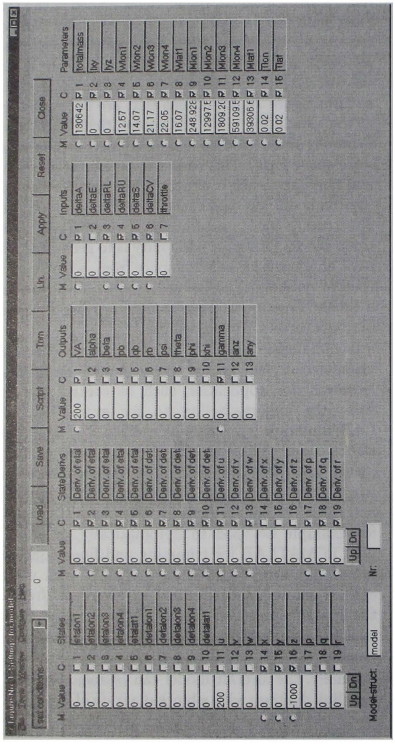


Figure 8: Screen shot of MATLAB trim GUI

the elevator. The pitch rate is sensed near the cockpit. Using the software described in ref. [19], which combines available symbolic and numeric computational software tools, we automatically generate a model in the form of a Linear Fractional Transformation (LFT) from the parametric Dymola model. The LFT, in which varying parameters are separated from the model and interconnected in a feedback path, is the standard form for μ-analysis. The LFT of the example aircraft model contains airspeed, structural eigenvalues and modal damping as varying parameters. With μ-analysis a worst-case flutter speed is computed in face of the uncertain structural parameters. The worst-case parameter combination is substituted into the Dymola generated simulink model, to verify the occurrence of flutter in a simulation. It turns out that the third symmetrical mode goes unstable, see figures 9, 10.

Since both the LFT model and simulation model have the same Dymola model as origin, this illustrates the advantage of single-source modeling capability (figure 7).

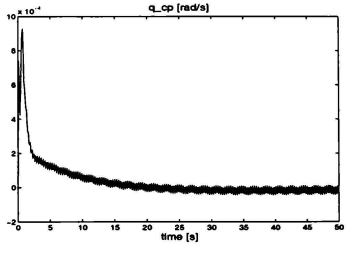


Figure 9: Pitch rate in cockpit: verification of flutter for worst model parameters

7. Conclusions

In this paper we have proposed a software implementation of integrated flight mechanics / aeroelastic aircraft models exploiting the features of object oriented modeling. Sub-models from different engineering disciplines can be implemented one-to-one

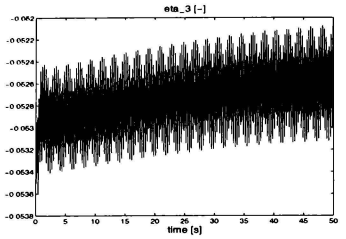


Figure 10: Gen. displacement of third symm. mode

into software objects, since we are able so specify the interconnections according to the way the objects physically interact.

The equations of motion form the core of the model implementation. The rigid simulation model is used as a basis, while the fidelity of the aeroelastic part may depend on the accuracy required by the application.

The implementation of an elastic aircraft model of limited complexity was demonstrated. Except for the aerodynamic model, other model components can be easily re-used in other aircraft models by adapting values in the object parameter window. At the moment, the fidelity of the aeroelastic model can only be adjusted via the number of modes. For more complex models this, for example, could also be done via the order of the rational function approximation of the unsteady aerodynamics, by implementing several aerodynamic models that are easily exchangable in the integrated aircraft model.

The package used for implementing the model, Dymola, is able to automatically generate simulation code for several engineering environments. The possibility to generate symbolic code facilitates parametric uncertainty modeling, since model parameters can be treated explicitly in e.g. a symbolic mathematical package. A great advantage is that the symbolic and the simulation model are generated from the same source. An example was given by generating an LFT model from the Dymola code to perform flutter analysis.

So far, the implementation is conceptual, since the complexity of the implemented aircraft model is low, and the model was readily at hand. Future work will involve actual modeling work and implementation of a more complex aircraft model.

Acknowledgment

The author wishes to thank Dieter Moormann from the same department for his support in all Dymola implementation work.

References

[1] E. Albano and W.P. Rodden. A Doublet-Lattice Method for Calculating Lifting Disturbances of Oscillating Surfaces in Subsonic Flows. *AIAA Journal*, 7(2):279–285, 1969. (Errata in AIAA Journal 7(11), November 1969, p.2192).

[2] P. Douglas Arbuckle, Carey S. Buttrill, and Thomas A. Zeiler. Simulation Model Building Procedure for Dynamic Systems Integration. *Journal of Guidance Control and Dynamics*, 12:894–900, 1989.

[3] Rudolf Brockhaus. *Flugregelung*. Springer-Verlag, Berlin Heidelberg, 1994.

[4] C.S. Buttrill, T.A. Zeiler, and P.D. Arbuckle. Nonlinear Simulation of a Flexible Aircraft in Maneuvering Flight. In *Proceedings of the AIAA Flight Simulation Technologies Conference*, Monterey, CA, August 1987. AIAA-87-2501.

[5] H. Elmqvist. Object-oriented modeling and automatic formula manipulation in dymola. In *SIMS '93, Scandinavian Simulation Society*, Kongberg, Norway, June 1993. available from `http://www.dynasim.se/publications.html`.

[6] Modelica Design Group. Modelica: Language design for multi-domain modeling. `http://www.modelica.org`.

[7] Integrated Systems Inc. *SystemBuild User's Guide, version 6*, 1998.

[8] M. Karpel and S.T. Hoadley. Physically Weighted Minimum State Unsteady Aerodynamic Approximation. Technical Report NASA-TP-3025, NASA, 1990.

[9] E. Levretsky. High Speed Civil Transport (HCST) Flight Simulation and Analysis Software Development. In *Proceedings of the 36th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, Jan 1998. AIAA-98-0173.

[10] Rick Lind and Martin J. Brenner. Robust Flutter Margin Analysis that Incorporates Flight Data. Technical Report TP-1998-206543, NASA, Dryden Flight Research Center, Edwards CA, 1998.

[11] The Math Works Inc. *Simulink User's Guide*, 1998.

[12] R.D. Milne. Dynamics of the Deformable Airplane (Part I and II). Technical Report R & M Number 3345, and AD-A953155, Her Majesty's Stationary Office, London, England, 1964.

[13] Dieter Moormann. GARTEUR Research Civil Aircraft Model (RCAM): Aircraft Physical Modeling Leading to Automatic Code Generation. Technical report, DLR-Oberpfaffenhofen, Institute for Robotics and System Dynamics, 1997. Hypertext document: `http://www.op.dlr.de/DD-DR-ER/research/flight/rcammod/node2.html`.

[14] D. Moormnann, P.J. Mosterman, and G. Looye. Object-oriented computational model building of aircraft flight dynamics and systems. *Aerospace Science and Technology*, 3:115–126, 1999. To appear early 1999.

[15] Martin Otter. *Objektorientierte Modellierung mechatronischer Systeme am Beispiel Geregelter Roboter*. PhD thesis, Fakultät für Maschinenbau der Ruhr-Universität Bochum, November 1993.

[16] K. Roger. Airplane Math Modeling Methods for Active Control Design. *AGARD*, (CP-228), August 1977.

[17] J. Schuler. *Flugregelung und aktive Schwingungsdämpfung für flexible Großraumflugzeuge - Modellbildung und Simulation.* PhD thesis, Institut für Flugmechanik und Flugregelung, Universität Stuttgart, 1997.

[18] A. Varga and G. Looye. Symbolic and numerical software tools for lft-based low order uncertainty modeling. In *Proc. CACSD'99 Symposium, Cohala Coast, Hawaii*, 1999.

[19] A. Varga, G. Looye, D. Moormann, and G. Grübel. Automated Generation of LFT-based Parametric Uncertainty Descriptions from Generic Aircraft Models. *Mathematical and Computer Modelling of Dynamical Systems*, 4(4):249–274, 1998. See also GARTEUR report TP-088-36 at `http://www.nlr.nl/public/hosted-sites/garteur/sum36.html`.

[20] Martin R. Waszak and Dave K. Schmidt. Flight Dynamics of Aeroelastic Vehicles. *Journal of Aircraft*, 25(6):563–571, June 1988.

[21] M.R. Waszak and D.K. Schmidt. A Simulation Study of the Flight Dynamics of Elastic Aircraft: Volume 1 - Experiment, Results and Analysis. Technical Report NASA CP-4102, NASA, December 1987.

[22] B.A. Winther, P.J. Goggin, and J.R. Dykman. Reduced Order Dynamic Aeroelastic Model Development and Integration with Nonlinear Simulation. In *Proceedings of the AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1694 – 1704, Los Angeles, April 1998.

DEVELOPMENT OF A FREE-FLIGHT SIMULATION INFRASTRUCTURE

Eric S. Miles
Paul C. Davis
Seagull Technology, Inc.
16400 Lark Ave., Los Gatos, CA 95032

David J. Wing
NASA Langley Research Center
Hampton, VA 23681

ABSTRACT

In anticipation of a projected rise in demand for air transportation, NASA and the FAA are researching new air-traffic-management (ATM) concepts that fall under the paradigm known broadly as "free flight". This paper documents the software development and engineering efforts in progress by Seagull Technology, to develop a free-flight simulation (FFSIM) that is intended to help NASA researchers test mature-state concepts for free flight, otherwise referred to in this paper as distributed air / ground traffic management (DAG TM). Under development is a distributed, human-in-the-loop simulation tool that is comprehensive in its consideration of current and envisioned communication, navigation and surveillance (CNS) components, and will allow evaluation of critical air and ground traffic management technologies from an overall systems perspective. The FFSIM infrastructure is designed to incorporate all three major components of the ATM triad: aircraft flight decks, air traffic control (ATC), and (eventually) airline operational control (AOC) centers.

INTRODUCTION

Over the past twenty years, increased demand for air travel has outpaced the design of the National Airspace System (NAS). Insufficient capacity, limited access, and excessive operating restrictions have led to significant increases in user costs and delays and an overall decrease in efficiency for all users [1]. Furthermore, the market demand for air travel is expected to increase several-fold in the coming decade. Recent technological advances now provide the opportunity to redesign the NAS to substantially increase its capacity to accommodate air traffic growth and to provide substantial operational flexibility to airspace users, while maintaining, and possibly improving, safety.

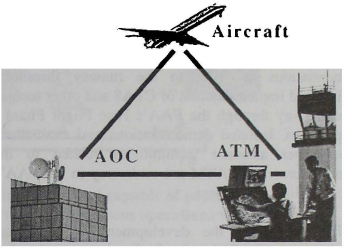


Figure 1: The FFSIM infrastructure is designed to incorporate all three major components of the ATM triad: aircraft flight decks, air traffic control (ATC), and airline operational control (AOC) centers.

A broad concept of NAS operations known as "free flight", proposed by the RTCA [1] in 1995, may provide these and other benefits through application of new technologies and procedures. The primary difference between today's direct-route-clearance approach and free flight will be the pilot's ability to operate the flight without the requirement to follow specific route, speed, and altitude clearances. In mature visions of free flight, the flight crew of properly equipped aircraft would be given the authority to maneuver at will while broadcasting their intentions to other system participants and maintaining responsibility for ensuring minimum separation standards are met. Aircraft without the equipment necessary for performing these functions would receive clearances from the ground through voice or data

Copyright © 1999 by The American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

link communications, and separation assurance responsibility for these aircraft would remain with the ground-based air traffic service provider (ATSP). This distribution of responsibility and authority will be referred to in this paper as distributed air / ground traffic management (DAG TM).

In recent years, the ATM community has been focused on the development of technologies and evolutionary improvements in efficiency and capacity based on the current mode of ground-based air traffic control. The Center-TRACON Automation System (CTAS) under development by NASA [2, 3] will provide air traffic controllers a set of tools to better manage scheduling and sequencing of aircraft into the terminal area with continuous updating to the runway threshold. Limited implementation of CTAS and other tools is underway through the FAA's Free Flight Phase I program. Isolated demonstrations and evaluations of other airborne technologies related to the advancement of ATM are underway in the FAA's Safe Flight 21 project.

In contrast to the development of individual airborne and ATM technologies, less attention has been given to date to establishing the overall system feasibility of mature-state free-flight concepts and the integration of essential underlying CNS components. The work documented in this paper leverages these past and existing airborne and ATM research endeavors while filling a void in the ATM research community by providing a mechanism for evaluating the system-level feasibility of free-flight operations and the integration of enabling technologies, including CNS components and decision support tools (DSTs), for both the flight crew and the air traffic service providers on the ground.

RELATED DEVELOPMENTS

Several recent and current development efforts are relevant to the FFSIM infrastructure development. The efforts are briefly described below, and the connection of each to FFSIM development or NASA research is identified. A thorough discussion of these endeavors is beyond the scope of this paper.

Ground ATM Decision Support Tools

NASA researchers have been developing and field-testing CTAS, a DST for ground-based air-traffic management [2, 3]. Core components of CTAS are leveraged in the FFSIM infrastructure to provide the situation display for the ATSP station. (The notion in free flight of distributing traffic management functions between the ground and the air leads to the re-designation of air traffic controller to air traffic service provider.) Inclusion of these CTAS core components allows for eventual expansion to include CTAS components such as the Traffic Management Advisor (TMA) for scheduling and En-route / Descent Advisor (E/DA) for ATSP decision support.

Flight Simulation Tools

NASA has developed a versatile PC-based simulation of a transport-category aircraft called "FastWin" (FMS-Autoflight Simulation Tools for Windows) [4]. FastWin includes an aircraft performance model, a Flight Management System (FMS) model, Control Display Unit (CDU), Mode Control Panel (MCP), and primary cockpit displays. FastWin is integrated into the FFSIM infrastructure as the core component of the research pilot station.

Logicon previously developed a software simulation tool entitled "Pseudo Aircraft Systems" (PAS) to serve as a target generation facility and pseudo-pilot station with a low-fidelity auto-pilot capability for CTAS development experiments [5]. The PAS software is used in the FFSIM infrastructure to manage traffic scenarios, to generate targets, and to provide the ability for a human operator to control multiple pseudo-pilots.

The National Aerospace Laboratory of The Netherlands (NLR) is currently conducting extensive ATM research activities through human-in-the-loop simulation. Notable simulation tools developed by NLR include TMX, a target-aircraft generator; AIRSIM, an interface to a single pilot station; and NARSIM, a distributed simulation effort aimed at investigating the European version of free flight [6]. A close working relationship has been established between Seagull and NLR to exploit lessons learned during their research and to accelerate FFSIM design and development. NASA and NLR are exploring a collaborative approach to free-flight research with activities starting in 1999.

Eurocontrol has also been very active in ATM research. Efforts relevant to FFSIM include: OASIS, a generic simulation infrastructure based on the Common Object Request Broker Architecture (CORBA); ESCAPE, a distributed air traffic control (ATC) simulation that uses OASIS; and FREER, a four-part research program that is

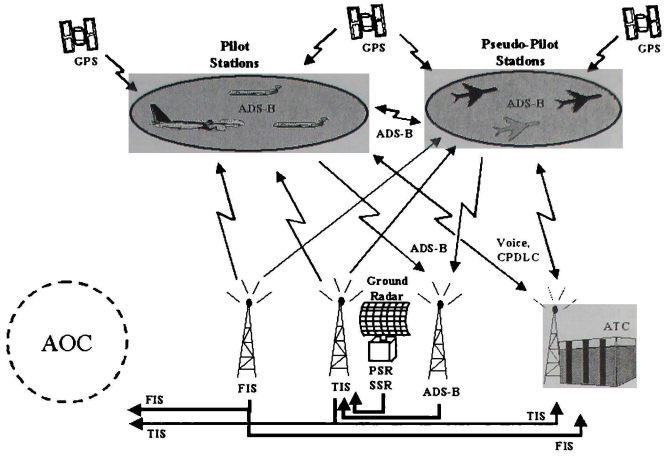


Figure 2: The FFSIM infrastructure is a distributed simulation tool capable of providing a comprehensive model of current and envisioned air-traffic-management operations with future communication, navigation, and surveillance capabilities.

simulating, developing, and flight-testing a prototype system for fully-autonomous aircraft operations.

Conflict Detection and Resolution

Numerous organizations have developed software algorithms for conflict detection and resolution (CD&R) [7]. Seagull Technology has developed a solution approach that can be applied in both ground and airborne contexts [8]. MIT has performed significant research in CD&R [9, 10]. Lincoln Laboratory at MIT developed a self-organizational approach for conflict resolution [11]. This last approach, combined with conflict detection based on aircraft state information and flight plans, was implemented as an initial decision-aiding prototype system in FFSIM for the flight deck and the ATSP by Lockheed Martin Aeronautical Systems [12].

CNS Modeling

MIT Lincoln Labs has extensively studied the automatic dependent surveillance – broadcast (ADS-B) concept [13, 14]. These theoretical and empirical studies of ADS-B, which include message-collision modeling, are leveraged in the FFSIM representation of the CNS infrastructure. NASA Glenn Research Center is currently developing high-fidelity parametric models for communication mechanisms such as Mode S, VHF Datalink (VDL) Modes 2, 3, and 4, etc. Many of these models will be incorporated into FFSIM to permit assessment of design requirements for free-flight application of these technologies.

FFSIM Infrastructure and Research Capabilities

The FFSIM infrastructure is unique in that it provides system-level integration and modeling of technologies vital for free flight with sufficient fidelity for assessing their impact on the feasibility of free-flight operations. FFSIM provides a human-in-the-loop environment where several research-subject pilots fly simulated aircraft through simulated airspace managed by one or more research-subject ATSPs. The human-subject pilots and ATSPs may have the assistance of prototype DSTs providing separation assurance and other navigation functions and may communicate by voice or digital communications. CNS technologies are modeled with moderate realism to allow the processes of communication, navigation, and surveillance to occur within the free-flight environment with realistic characteristics and limitations. Additionally, the simulated airspace may be populated by target aircraft controlled by

human participants responding to voice clearances issued by the ATSPs.

FFSIM will provide research capabilities in several areas. First, it will allow the assessment of CNS infrastructure design characteristics with respect to free-flight operations. Performance standards are emerging for technologies vital to free flight without the benefit of requirements generated under free-flight operating conditions. Data generated by FFSIM will begin to shed light on these requirements. Second, FFSIM will provide a means for assessing human performance and procedures to the extent appropriate in a medium-fidelity simulation environment. The type and presentation of information required by the human participants to maintain awareness and to perform the necessary duties of their stations can be assessed. Third, parametric controls on important factors such as wind-prediction accuracy will allow determination of trajectory-prediction requirements for conflict detection. Fourth, FFSIM will permit assessment of CD&R feasibility in both level and transition phases of flight and in a variety of airspace-constrained and schedule-constrained scenarios. In particular, the ability of aircraft transitioning to terminal-area operations to maintain separation while adhering to operational air traffic scheduling is critical for assessing feasibility of free flight since the vast majority of U.S. domestic airspace contains aircraft in this mode of operation. Additionally, this ability must be assessed in the presence of aircraft with mixed levels of equipage (and therefore varying ability to self-separate).

The current scope of FFSIM development focuses on the ATSP and aircraft flight deck aspects of DAG TM operations. However, the FFSIM infrastructure is designed to facilitate the incorporation of a future AOC presence, as shown in Figure 1. The FFSIM infrastructure allows participants to emulate the roles of pilots and pseudo-pilots, ground-based controllers, and, eventually, airline dispatchers.

FFSIM Infrastructure Components

The FFSIM infrastructure is a distributed simulation tool comprised of many components. The majority of these components are depicted in Figure 2. Development emphasis to date has focused on the aircraft flight deck, the ATSP, and the CNS infrastructure components. Modeling of the AOC and information sharing between the AOC and the flight deck and in the ATSP are planned for future phases of FFSIM development. A brief description of each FFSIM component follows.

Pilot Stations

The airborne portion of the FFSIM infrastructure is comprised of two types of flight-deck environments. The first environment, the "pilot station", provides an opportunity for several humans (i.e. research-subject pilots) to each control a single aircraft in real time with a reasonably high level of interface fidelity. The pilot station is implemented on one or two PC workstations and is comprised of an aircraft performance model, an interactive flight management system (FMS), a primary-flight display (PFD), a navigation display (ND), a mode control panel (MCP), communications, navigation, and surveillance equipment. These components, with the exception of the CNS equipment, are represented by the NASA-developed FastWin software package, which has been integrated with the FFSIM infrastructure. Figure 3 depicts the FastWin PFD and ND used for the FFSIM pilot station. The navigation display will be evolved into an integrated multifunction display (MFD) that will include navigation, flight plan, traffic, weather, and other data, in a user-configurable, layered format.

The CNS equipment models the hardware components required by the pilot station. Components include a voice transceiver for two-way verbal communication with the ATSP (and eventually the AOC), a transceiver for two-way digital controller-pilot data link communications (CPDLC) with the ATSP (and eventually the AOC), a global positioning system (GPS) receiver, an ADS-B transceiver, and receivers for Traffic Information Services (TIS) data and Flight Information Services (FIS) data.

The pilot stations may be configured to contain an integrated DST that accepts information regarding its own airplane, other traffic, and NAS status. The FFSIM infrastructure allows an integrated DST to output recommended conflict-free flight trajectories that adhere to specified constraints. The development of such DSTs is not within the scope of the FFSIM infrastructure development.

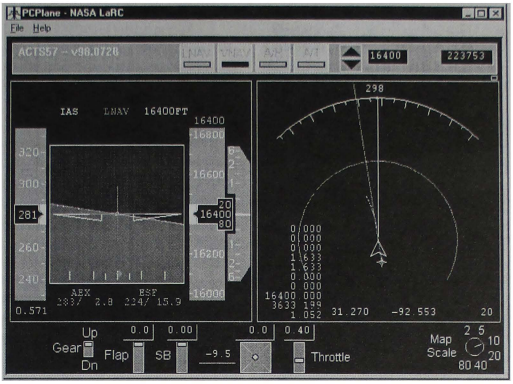


Figure 3: The FastWin PFD and Navigation Displays are integral parts of the FFSIM research pilot station. (Courtesy of NASA)

Pseudo-Pilot Stations

The second flight-deck environment in the FFSIM infrastructure is the pseudo-pilot station, which allows several humans (i.e. pseudo pilots) to each control multiple aircraft (up to a dozen each) in real time with a low level of interface fidelity. The pseudo-pilot capability enables FFSIM traffic scenarios to be populated with many aircraft operated by just a few humans.

Similar to the pilot station, the pseudo-pilot station contains an auto-pilot capability, CNS equipment, and possible instances of DSTs similar to those that would serve the research pilot stations. No individual cockpit displays such as the PFD and MFD are included in the pseudo-pilot station. The pseudo-pilot station auto-pilot and control interface are represented by the PAS software. The pseudo-pilot CNS equipment is essentially the duplicate of that employed by the pilot stations and is configured to serve multiple aircraft.

Air Traffic Service Provider Stations

The FFSIM ground infrastructure includes an ATSP component that allows human subjects to regulate and separate air traffic according to both current and future (envisioned) operating procedures for ATM. The central ingredient of the FFSIM ATSP component is the NASA-developed CTAS software. Figure 4 depicts the CTAS horizontal situation display known as the "pGUI". Multiple ATSP stations representing adjacent sectors can operate simultaneously, allowing research to pursue multi-sector coordination issues.

The FFSIM infrastructure provides both voice and digital communications between the ATSP and all appropriately equipped aircraft allowing for simulation of both current and various future concepts for air / ground interaction.

The ATSP component of the FFSIM infrastructure is designed to incorporate ground-based DSTs. However, the development of such DSTs is not within the scope of the FFSIM infrastructure development.

Communication, Navigation and Surveillance Components

The comprehensive modeling of CNS components at a level of fidelity sufficient for concept feasibility assessment and operational requirements definition is a major focus of this development effort. The ability to selectively increase the fidelity of any component has been designed into the FFSIM infrastructure. Each of the CNS infrastructure components is briefly described below.

Automatic Dependent Surveillance – Broadcast

ADS-B messages are broadcast by all aircraft equipped with an ADS-B transceiver and received by all similarly-equipped aircraft and the ATSP.

Both the broadcast rate and the nominal operating range for ADS-B are scenario parameters that can be configured by the researcher.

Each ADS-B message will contain, at a minimum, the following basic message content: computer identification number, timestamp, aircraft type, position source, latitude, longitude, altitude source, altitude, ground speed, ground-track direction, and vertical speed. The level of intent (i.e., the number of intended future trajectory change points) to be included in the ADS-B message can be configured by the researcher.

The FFSIM infrastructure modeling of ADS-B will be initially consistent with Mode-S data communication, yet will not preclude the option of future modeling of other viable technologies capable of implementing ADS-B.

Ground-Based Surveillance

Radar is the centerpiece surveillance system that enables current air traffic control. It is projected to continue to play a vital, albeit diminished, role as operations evolve toward free flight. The FFSIM infrastructure modeling of ground-based surveillance will include both primary and secondary surveillance radar (PSR and SSR).

The PSR model will include the Air Route Surveillance Radar (ARSR), used in the en-route airspace, and the Airport Surveillance Radar (ASR), used in the TRACON airspace [15]. The SSR model will include the interrogation-reply interaction process of secondary surveillance, including the envisioned ground-initiated communication broadcast (GICB) concept detailed in [16]. In the GICB concept, the transponder reply to ground interrogation will include the most recent ADS-B message of the interrogated aircraft, permitting independent verification of ADS-B messaging on the ground.

Traffic Information Services

As depicted in Figure 2, TIS provides a compilation of ground-received ADS-B message data and PSR/SSR data that is broadcast to all equipped aircraft, the ATSP, and the AOCs. The TIS concept allows aircraft that are equipped with a TIS receiver, but not with an ADS-B transceiver,

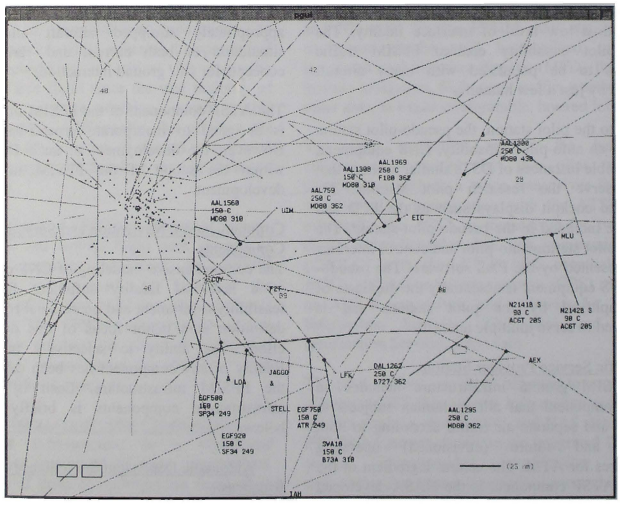


Figure 4: The CTAS pGUI controller display, shown here for ZFW, comprises the main controller display for the FFSIM infrastructure. (Courtesy of NASA)

to receive the position, velocity and intent content of neighboring aircraft [17]. For aircraft equipped with ADS-B, TIS provides a redundant source of traffic information. The TIS broadcast rate and nominal-operating range are scenario parameters that can be configured by the researcher.

Flight Information Services

FIS is a broadcast of weather, wind, special-use airspace (SUA), and other pertinent information from the ground to equipped aircraft [18]. A variety of FIS sources, such as NEXRAD weather radar, METARs, etc., will provide the FIS data content.

The FFSIM infrastructure modeling of FIS will be initially consistent with VDL Mode-2 technology, a likely candidate for FIS realization. Again, other viable technologies capable of implementing FIS will not be precluded. The FIS broadcast rate and nominal-operating range are scenario parameters that can be selected by the researcher.

Controller-Pilot Data Link Communications

CPDLC allows for digital communications between the ATSP station and the pilot and pseudo-pilot stations. A graphical interface will be implemented for both the pilot and ATSP stations to facilitate the use of CPDLC [19].

CPDLC is envisioned for many applications in current and future ATM operations, including the up-link of clearances, route deviations, and new trajectories from the ATSP to the pilot. One application of CPDLC, important in the DAG TM concept, is the digital up-link of conflict resolution maneuvers from a ground-based DST to the pertinent aircraft. Pilots will use CPDLC to respond to and possibly request ATSP instructions, as well as inform ATSP of changes to the current flight trajectory.

The FFSIM infrastructure modeling of CPDLC will be initially consistent with VDL Mode 3 technology, a likely candidate for realizing CPDLC. Other communication technologies for implementing CPDLC will not be precluded.

Global Positioning System Navigation

The position and velocity data contained in each ADS-B message is provided by a GPS satellite navigation receiver model. The FFSIM infrastructure includes a stochastic error model of selective availability (SA) for the basic navigation signal [20, 21]. Modeling of wide-area and local-area augmentation system (WAAS and LAAS) accuracy levels are planned as future enhancements.

Airline Operational Control

The third and final entity of the ATM operations triad depicted in the lower left of Figure 1 is the AOC facility. Although the initial development effort has focused on the flight deck and ATSP aspects of ATM operations, the FFSIM infrastructure is designed to facilitate incorporation of an AOC presence in the future, complete with all the communication channels between the AOC, the flight decks, and the ATSP.

SOFTWARE IMPLEMENTATION

This section describes details of the FFSIM infrastructure software.

CORBA

The FFSIM infrastructure employs the Common Object Request Broker Architecture (CORBA) [22]. CORBA is a software industry standard for facilitating communication between objects in distributed applications. An important aspect of CORBA is the interface definition language (IDL). The IDL allows language independent software interfaces to be specified. An interface specified in IDL is compiled into code stubs for a specified target language, e.g. C++ or Java. A programmer creating a CORBA server must implement the skeleton methods in the generated code stubs. To create a client to the server, a programmer simply needs to create a proxy to the server by using the generated code.

CORBA was chosen over a more traditional TCP/IP approach because of its language-independent, object-oriented, and standardized nature. It was also felt, and later substantiated, that CORBA would be a faster and less error prone method for developing inter-process communications. Eurocontrol's successful implementation of CORBA for ATM simulations also contributed to the decision.

CORBA Services

The CORBA specification outlines standard services that are intended to provide generic support to any CORBA system. FFSIM is designed to leverage some of these services.

The CORBA Naming Service is used to store object names in a repository. Conceptually it is similar to the white pages of a telephone directory.

FFSIM uses the Naming Service to store names of objects that represent individual aircraft components as well as the names of simulation services. The object names exist in the Naming Service repository even when the simulation is not active. This allows a client to search the Naming Service and retrieve a specific object. If the server that contains that object is inactive the object will be automatically launched.

The CORBA Event Service is used to de-couple the clients from the servers. This service allows servers to publish specific types of events and clients to subscribe to those events. The server supplying the event does not need knowledge of the client consuming the event. A new service called the Notification Service extends the functionality of the Event Service and provides "quality of service" delivery of events. While currently not used in FFSIM, the Event Service or its replacement the Notification Service, will be a critical part of the future infrastructure.

The Time Service is another CORBA service that may be incorporated into FFSIM. The Time Service provides distributed time management for a CORBA system. Unlike the Naming, Event, and Notification services a commercial version of the Time Service is not available. Any Time Service created for FFSIM will also need to be extended to include simulation time management.

Component Integration

The CTAS, PAS, and FastWin software all handle external communication using TCP/IP sockets. These applications are representative of a large number of existing components that do not use CORBA, but may require integration into FFSIM. There are two approaches for integrating existing TCP/IP components into FFSIM.

The first approach is to add a CORBA interface to the existing component. The advantage of this approach is that the application will be completely "plugged" into the simulation and will be able to easily take advantage of any services provided by other CORBA components. The disadvantage of this approach is that it may require extensive modifications, depending on the organization of the original source code.

The second approach is to create a new process that can communicate with the non-CORBA application using an existing TCP/IP interface and with the rest of the simulation using CORBA. The advantage of this approach is that, depending on the state of the existing application, little if any source code modifications are required. This may also be the only option if source code cannot be modified. The disadvantage of this approach is that it adds a layer of overhead and can be more time consuming to implement than directly integrating CORBA.

The FFSIM infrastructure uses the second approach for integrating FastWin. The FastWin application contains three separate processes that would potentially need to be modified to integrate with CORBA. Because FastWin is a research tool, NASA is continuously updating it. For these reasons it was decided that the best approach was to create a single CORBA to TCP/IP bridge and minimize changes to the existing FastWin code.

The integration of PAS into FFSIM was also accomplished using a CORBA to TCP/IP bridge. The primary reason for using a bridge in this instance was the proprietary nature of the PAS source code prevented direct modifications.

Interfaces

The FFSIM infrastructure interfaces were created using the CORBA IDL. The IDL provides greater flexibility than a traditional byte or text message sent over a TCP/IP socket. The CORBA interfaces create a simulation databus. Figure 5 illustrates the CORBA communication between components within the pilot station. Using the existing FFSIM interfaces a new component could easily be added.

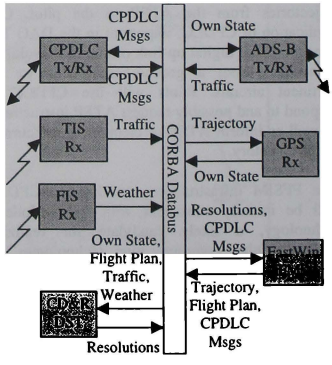


Figure 5: Pilot Station Component Communication.

Message Filtering

In some cases the FFSIM interfaces provide access to more data than is realistic or even necessary. However, this makes it easier to examine future concepts where this additional data may be needed.

One strategy the FFSIM infrastructure uses is client-side filtering. For example, an airborne CPDLC transceiver broadcasts and receives datalink messages to and from an ATSP CPDLC transceiver on the ground. The probability of a successful transmission is determined by many factors including the distance between transceivers. The receiving CPDLC transceiver determines if a message is successfully received based on the distance to the transmitting transceiver. In order to accomplish this the receiver must know information about the transmitter that may not be contained in the message, e.g. its location. The solution is to provide this information in a message header that is only used by the CPDLC transceiver and not passed along to any other ground or airborne applications. It is left to the individual implementation to decide what to do with the information in the message header. In the case of the CPDLC transceiver, the message receiver would use the location of the transmitter to determine if the transmitter is out of range. If it does determines that the transmitter is out of range the message is logged as being dropped and then ignored as if it was never received. The concept of client-side message filtering allows the component developer to decide what information should be provided to the component.

Deployment

FFSIM is a distributed multi-platform workstation based simulation that currently runs on computers with Solaris and Windows NT operating systems. The nominal FFSIM configuration consists of three pilot stations, three pseudo-pilot stations, two controller stations, and a simulation server station.

The CTAS ATSP software and the PAS pseudo-pilot software run on Solaris. The FastWin software, the CNS equipment, and the simulation infrastructure run on Windows NT.

CONCLUSIONS

A medium-fidelity workstation-based simulation of airspace operations is under development and has reached its first phase of completion. The Free Flight Simulation (FFSIM) was designed specifically to permit exploration of a proposed future mode of operations known as "free flight". In mature visions of free-flight operations, both the authority to approve changes in trajectory and the responsibility to ensure regulatory separation from other aircraft are distributed between aircraft flight crews and ground-based air traffic service providers.

The FFSIM infrastructure provides a simulated airspace environment complete with communication, navigation, and surveillance component models. Human operators participate in the simulation by controlling aircraft as pilots or pseudo pilots, managing airspace as air traffic service providers, and interacting with the other human operators in the simulation while using simulated technology appropriate for their role in free-flight operations.

The simulation was designed as a research tool to permit investigation of a variety of research issues and the development of technology requirements for free-flight operations. Examples of research areas appropriate for investigation in FFSIM include human actions and interactions in a distributed air/ground traffic management environment, procedures for free-flight operations, time horizons, feasibility of aircraft to autonomously maintain regulatory separation while efficiently meeting scheduled arrival constraints, and many other human-factors and engineering-related research issues. Examples of technology design appropriate for FFSIM usage include determining requirements for surveillance range and latency, intent broadcast and inference, wind and trajectory prediction accuracy, conflict detection and resolution algorithms, and essentially any variable under parameter control in FFSIM.

Much progress has been made in improving capacity and efficiency in the current mode of NAS operations through efforts such as CTAS. Free flight offers the possibility for significant increases in the ability of the NAS to accommodate air traffic growth and significant increases in operational flexibility for airspace users, while maintaining and possibly improving safety. A simulation capability, FFSIM, has now been developed that will serve as a research tool in defining technology requirements and determining feasibility of the free-flight operational concept.

REFERENCES

[1] *Report of the RTCA Board of Directors' Select Committee on Free Flight*, RTCA, January 1995.

[2] Erzberger, H., Davis, T. J., and Green, S., "Design of Center-TRACON Automation System."

AGARD Guidance and Control Symposium on Machine Intellignce in Air Traffic Management, Berlin, Germany, May 1993.

[3] Denery, D. G. and Erzberger, H., "The Center-TRACON Automation System: Simulation and Field Testing," *Proceedings of the advanced Workshop on ATM (ATM 95)*, sponsored by the National Research Council of Italy, Capri, Italy, Oct. 2-6, 1995. (Also published as NASA Technical Memorandum 110366, August 1995).

[4] Palmer, M. T., Abbott, T. A., Williams, D. H., "Development of Workstation-Based Flight Management Simulation Capabilities within NASA Langley's Flight Dynamics and Control Division", *Ninth International Symposium on Aviation Psychology*, Columbus, OH, April, 28-May, 1, 1997.

[5] *Pseudo-Aircraft Systems (PAS) Users's Manual*, Revision 4.2, Logicon/Syscon/Syre, November 1997.

[6] NLR Web Site, www.nlr.nl, and personal discussions.

[7] Kuchar, J. K. and L. C. Yang, "Survey of Conflict Detection and Resolution Modeling Methods", AIAA-97-3732, AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, August 11-13, 1997.

[8] Krozel, J. and Peters, M. "Conflict Detection and Resolution for Free Flight", *Air Traffic Control Quarterly*, Vol. 5, No. 3, pp. 181-212, 1997.

[9] Yang, L. C., and J. K. Kuchar, "Prototype Conflict Alerting Logic for Free Flight", AIAA Journal of Guidance, Control, and Dynamics, Vol. 20, No. 4, July-August 1997.

[10] Kuchar, J. K. and L. C. Yang, "Incorporation of Uncertain Intent Information in Conflict Detection and Resolution", 36th IEEE Conference on Decision and Control, San Diego, CA, December 10-12, 1997.

[11] Eby, Martin S., "A Self-Organizational Approach for Resolving Air Traffic Conflicts", The Lincoln Laboratory Journal, Vol. 7, No. 2, 1994.

[12] "User Instruction Manual for NASA Research Task Order (RTO) –5(A) Free Flight Simulation Components Development", Lockheed Martin Corporation, January 1999.

[13] Orlando, V. A., Harmon, W. H., *GPS-Squitter Capacity Analysis*, MIT Lincoln Laboratory, Project Report ATC-214, 1994.

[14] *Minimum Aviation System Performance Standards for Automatic Dependent Surveillance Broadcast (ADS-B)*, RTCA, SC-186, DO-242, 1998.

[15] M. S. Nolan, *Fundamentals of Air Traffic Control*, 2nd Ed., Wadsworth, Inc., 1994.

[16] *National Airspace System Architecture-Version 4.0*, Federal Aviation Administration, ASD-1, January 1999.

[17] *Minimum Operational Performance Standards for Traffic Information Service (TIS) Data Link Communications*, RTCA, SC-169, DO-239, 1997.

[18] *Operational Concepts for Data Link Applications of Flight Information Services (FIS)*, RTCA, SC-169, DO-232, 1996.

[19] *Minimum Operational Performance Standards for ATC Two-Way Data Link Communications*, RTCA, SC-169, DO-219, 1993.

[20] *Global Positioning System: Theory and Applications*, Vol. 1, AIAA, Vol.163, pp 606-608, 1996.

[21] "Global Positioning System Standard Positioning Service Signal Specification," 2nd Ed., *DoD GPS/JPO*, p. B-18, June 2, 1995.

[22] OMG Web Site, www.omg.org.

AERODYNAMIC DATABASE ISSUES FOR MODELING CLOSE FORMATION FLIGHT

James H. Myatt* and William B. Blake†
Air Vehicles Directorate
Air Force Research Laboratory
Wright-Patterson AFB, OH

ABSTRACT

Experimental data for the trail aircraft in a two-ship formation are presented. Aerodynamic force and moment data show that the drag of the trail aircraft is reduced for a given lift when the aircraft are properly positioned. The data are analyzed from the perspective of simplifications to the aerodynamic database needed for adequate simulation of close formation flight.

NOMENCLATURE

C_D = drag coefficient, D/qS

C_L = lift coefficient, L/qS

C_l = rolling moment coefficient, ℓ/qSb

C_m = pitching moment coefficient about 1/4 m.a.c., $m/qS\bar{c}$

C_n = yawing moment coefficient, n/qSb

C_Y = side force coefficient, Y/qS

α = angle of attack, deg

β = sideslip angle, deg

ζ = vertical distance of the trail aircraft above the lead aircraft, normalized by span

η = lateral distance of the nose of the trail aircraft to the right of the nose of the lead aircraft, normalized by span

ξ = longitudinal distance of the nose of the trail aircraft behind the nose of the lead aircraft, normalized by span

*Aerospace Engineer, Member AIAA.
†Aerospace Engineer, Associate Fellow AIAA.
This paper is declared a work of the U. S. Government and is not subject to copyright protection in the United States.

INTRODUCTION

Any flight condition involving two or more vehicles where the aerodynamic interference between the vehicles must be considered is defined to be close formation flight. Current military missions that involve close formation flight include aerial refueling, minimum interval takeoff, and towing. Long-range ferry missions are typically flown in formations with the aircraft spaced vertically to avoid each other's wakes. These are not considered "close formations" from the perspective of the current paper. Wake vortex encounters, which are of high interest to the civilian community, are an outcome of unintended close formation flight.

One possible future application of close formation flight would be to increase range (via induced drag reduction) by flying aircraft in patterns similar to those flown by large flocks of birds. A recent wind tunnel test[1] has shown the drag benefit is real, and analytical studies[2,3,4] predict the benefit increases as additional aircraft are added to the formation.

Simulation of close formation flight can involve aerodynamic databases much larger than those associated with simulation of a single vehicle. Requirements for the scope of wind tunnel tests also become very large. If three or more vehicles are to be simulated, the size of the database or required tests may become prohibitive. This paper will examine the aerodynamic interference effects from a recent wind tunnel test of a two-ship formation and develop possible simplifications to the form of the aerodynamic math model.

TEST SETUP

The test (see Fig. 1) was conducted in the Langley Full-Scale Wind Tunnel, which is operated by Old Dominion University. The tunnel has an open test section 30 ft high by 60 ft wide. Testing was done at a dynamic pressure of 5 lb/ft^2. A 1/10 scale F-18 class aircraft was tested behind a wing of identical size and planform mounted on a

circular body. Forces and moments were measured on the trail aircraft using a six component internal balance. Pressure data were also obtained but will not be discussed in this paper. The lead wing-body was mounted on a carriage that permitted the relative distance between the two vehicles to be varied over a range of positions, as shown in Table 1. The lead aircraft was fixed at ten degrees angle-of-attack and zero degrees sideslip, and the trail aircraft was tested at zero, five and ten degrees angle-of-attack and at zero and +/- five degrees sideslip.

The trail aircraft was sting mounted, with the center of rotation of the sting aft of the model. This meant that there was a vertical displacement between the two vehicles as the angle of attack on the trail vehicle was increased. Similarly, a lateral displacement occurred when the trail aircraft was tested in a sideslip condition. The relative heights between the vehicles were selected so that a nearly constant vertical position could be maintained for more than one angle-of-attack. This is critical for the assessment of what parameters are independent of the trail aircraft angle-of-attack.

Only the largest downstream spacing of 2.25 spans (see Fig. 2) will be examined in this paper. Here, the effect of upwash from the trail aircraft on the lead is minimized, allowing the assumption that the lift on the lead vehicle is constant. In addition, the smaller spacing (1.50 span) is less realistic from an operational point of view due to safety concerns (collision avoidance).

EXPERIMENTAL RESULTS

Baseline Aircraft

Force and moment data for a single aircraft (no interference effects) are shown in Fig. 3. Sideslip effects on the longitudinal forces and moment are minimal. Linear and quadratic fits to lift and drag, respectively, show that the aircraft exhibits the expected behavior for the angle-of-attack range tested. The lateral/directional data show significant sideslip effects, as expected.

Drag Reduction

The drag reduction realized by a trail aircraft flying in the same horizontal plane and 2.25 spans downstream of the lead aircraft at a lateral separation of one span so that the wing tips are aligned, *i.e.*, (ξ,η,ζ) = (2.25,-1.00,0.00), is illustrated in Fig. 3b. The trail aircraft at zero side slip experiences an 11.1% decrease in drag for $C_L = 0.64$ and a 16.9% reduction for $C_L = 0.31$ compared to a single aircraft at zero sideslip. This reduction is due to the rotation of the lift vector of the trail aircraft forward due to upwash from the lead aircraft, thereby reducing the induced drag.[2] This dramatic reduction in drag can be used to decrease fuel consumption and/or extend range for aircraft flying in formation. However, this drag reduction is for an untrimmed trail aircraft, and control deflections needed to maintain the trail aircraft position may reduce the benefit of close formation flight. Nevertheless, it will be shown later that an optimum location can be found that provides significant drag reduction while requiring small control surface deflections to trim the aircraft.

Relative Position Effect

The increments to the forces and moments on a trail aircraft flying at angles of attack of 5° and 10° and zero sideslip are shown versus lateral location η for several vertical locations ζ in Figs. 4 and 5. The trail aircraft is 2.25 spans downstream of the lead aircraft in all cases. Although not shown, the data for the 1.50 span downstream spacing were very similar to the 2.25 span case. More data are needed at larger downstream spacing intervals to determine the sensitivity to this parameter. It is important to note that the lines connecting the data are not intended to indicate trends between points; they are included only to aid the reader in identifying test conditions.

It is clear that the forces and moments are highly nonlinear functions of both lateral and vertical spacing. As expected, the interference effects are greatest when the trail aircraft is flying in nearly the same horizontal plane as the upstream aircraft and tend to zero as the trail aircraft moves away from the lead aircraft in either the vertical or lateral direction. Fig 4b shows that the trail aircraft is stable with respect to vertical position when it is above the lead vehicle, and is unstable when below the lead vehicle.

For a trail aircraft location near (ξ,η,ζ) = (2.25,±1.00,0.00), a slight increase in drag results. However, this penalty is far outweighed by the corresponding increase in lift, allowing the trail aircraft to fly at a lower angle of attack, thereby achieving an overall decrease in drag.

Reference values for selected control surface deflections of 10° for an F-18 class aircraft are shown in Table 2 for C_l, C_m, C_n, and C_Y. Comparisons of the increments in Figs. 4c-4f and 5c-5f show that the largest effect is on rolling moment. This is due to the vortex wake roll-up

behind the lead aircraft. For a trail aircraft flying at $(\xi,\eta,\zeta) = (2.25,-0.50,0.00)$, *i.e.*, flying in the same horizontal plane with the nose of the trail aircraft directly behind the wing tip of the lead aircraft, a downwash is induced on the inboard side (with respect to the lead aircraft) of the trail aircraft, and an upwash is induced on the outboard side. This induces a large positive rolling moment that tends to force the trail aircraft to roll toward the lead aircraft. For a trail aircraft flying at $(\xi,\eta,\zeta) = (2.25,-1.00,0.00)$, the entire trail aircraft experiences an upwash, with the largest effect on the inboard wing, thereby inducing a negative rolling moment of lesser magnitude than for $\eta = -0.50$. While large rolling moments, compared to the reference value of 0.01 for a 10° aileron deflection on an F-18 class aircraft (indicated by the solid grid-lines in the figure), exist for some spanwise locations of the trail aircraft, the induced rolling moment near $\eta = -1.00$ is small. This indicates that it is possible to trim the aircraft at the location that gives the maximum benefit due to induced drag reduction without incurring a large penalty due to the control surface deflection.

Induced pitching moment, shown in Figs. 3d and 4d, is small compared to the reference value. Induced yawing moment and side force are significant, but no large control surface deflections are needed to trim the aircraft at the desired location.

Sideslip Effect

The effect of sideslip on the forces and moments is illustrated in Figs. 6 and 7 where the increments due to the interference are plotted for angles of attack of 5° and 10°, respectively. Due to some assymetries in the baseline data (no lead aircraft), it was necessary to make adjustments to the baseline data in order to determine the increments. The baseline drag for $\alpha = 10°$ and $\beta = 5°$ was increased to equal the drag for $\alpha = 10°$ and $\beta = -5°$, providing symmetry with respect to sideslip angle for this longitudinal datum and more nearly mimicking the increase in drag for both positive and negative sideslip angles observed at angles of attack of 8° and 12° (see Fig. 3). Likewise, lift and rolling moment were adjusted at $\alpha = 10°$ by using the data taken at $\alpha = 8°$ and 12° as a guide.

HASC95[5], a planar vortex lattice code, was used to predict the effect of spanwise spacing on the wake induced forces and moments on the trail aircraft. The lead configuration was modeled as a flat plate with 450 elements. The trail aircraft was modeled using 1066 elements and included camber, twist, and vertical panels for the fuselage and tails. Even spacing was used for elements in the spanwise direction and cosine spacing was used for the chordwise elements. Fig. 2 shows the lattices of the two configurations. Care was taken to ensure correct alignment of vortex filaments and control points when the aircraft overlapped in the lateral direction. Predictions for the wake-induced lift and rolling moment at zero sideslip are included with the data in Figs. 6 and 7. Except for overpredictions of the peak values, good agreement is found with the data, including the data at nonzero sideslip. This indicates that for this configuration, sideslip of the trail aircraft can be removed as an independent variable with respect to the wake induced increments.

Angle of Attack Effect

The effect of angle of attack on the force and moment increments is shown in Fig. 8. Increasing the angle of attack increases the wake-induced drag. Considering the effect of differential induced drag on the port and starboard wings, one would expect the wake induced yawing moment to be a function of angle of attack. The data show almost no variation, so the measured increments are presumably due to forebody and vertical tail effects. The remaining forces and moments are only slightly affected by changes in angle of attack. This is probably because these are primarily affected by the strength of the vortex system from the lead vehicle, whose lift (or angle of attack) was held fixed. Although the lift increment on the trail vehicle does not vary with angle of attack, its total lift does. This is the source of the increase in the induced drag increment. For this configuration, all of the wake-induced forces and moments except drag can be modeled without using angle of attack of the trail aircraft as an independent variable.

Effect on Stability

Figure 8 also illustrates the effects of close formation flying on the stability of the trail aircraft. Examination of the yawing moment data for the baseline aircraft in Fig. 3e shows that C_{n_β} is positive, with a value of 0.0016 at $\alpha = 5°$. The increments shown in Fig. 8 exhibit stabilizing characteristics at most conditions and have a maximum destabilizing increment an order of magnitude less than the baseline stability derivative.

The dihedral effect, with C_{l_β} = -.0016 at α = 5° for the baseline aircraft (see Fig. 3c), is diminished significantly by the induced velocity field encountered by the trail aircraft for sideslip angles that move further into the wake of the lead aircraft, but stability is retained. The effect on longitudinal stability, characterized by C_{m_α}, is an order of magnitude less than the magnitude of the stability of the baseline aircraft.

IMPLICATIONS FOR MODELING

Most simulations of close formation flight involve investigation of wake vortex encounters. Strip theory is typically used to predict the wake-induced increments on the trail vehicle. In the linear lift regime, this means that the wake induced forces and moments are considered independent of angle of attack and sideslip. The force and moment parameters considered range from only rolling moment[6] to rolling moment and lift[7] to all six parameters.[8] Only one or two relative positions (considered to be worst case), or an assumed wake penetration path are modeled. The data from this paper support many of these simplifications. The high sensitivity of the wake-induced forces and moments to relative position indicates that this should be accurately modeled.

CONCLUSIONS

Data from a wind tunnel test of two aircraft in close formation flight have been examined from the perspective of simplifying aerodynamic math models needed for simulation. For the configuration studied, the following conclusions are drawn:

1) Except for drag, wake induced forces and moments are effectively independent of the angle of attack of the trail aircraft.
2) Wake induced forces and moments are effectively independent of the sideslip angle of the trail aircraft. Together with (1), this implies that the effects of angle of attack and sideslip of the trail aircraft on the wake induced force and moment increments may be ignored in modeling and simulation.
3) Wake induced forces and moments are strong functions of the relative lateral and vertical positions of the lead and trail aircraft.
4) Additional data are needed regarding the effects of the attitude of the lead aircraft, roll attitude of the trail aircraft and the relative longitudinal position of the lead and trail aircraft.
5) Additional data are needed to determine the effects of control surface deflection, control required for trim, and the effect of control deflections on the induced drag benefit.
6) Additional data are needed near the optimum location of the trail aircraft (near wing tips aligned) to more fully populate the location data and better determine where the trail aircraft should fly to obtain the maximum benefit.

ACKNOWLEDGMENTS

The data were collected by Bihrle Applied Research, Inc. under SBIR contract F33615-98-C-3608. The efforts of Mr. David Gingras are gratefully acknowledged.

REFERENCES

[1]Gingras, D.R., "Experimental Investigation of a Multi-Aircraft Formation," AIAA 99-3143, presented at the 17th AIAA Applied Aerodynamics Conference, Norfolk, Virginia, June 1999.

[2]Blake, W. and Multhopp, D., "Design, Performance and Modeling Considerations for Close Formation Flight," AIAA Paper 98-4343, presented at AIAA Atmospheric Flight Mechanics Conference, Boston, MA, Aug. 1998.

[3]Maskew, B., "Formation Flying Benefits Based on Vortex Lattice Calculations," NASA CR-151974, May 1977.

[4]Hummel, D., "Aerodynamic Aspects of Formation Flight in Birds," *Journal of Theoretical Biology*, Vol. 104, pp. 321-347, 1983.

[5]Albright, A.E., Dixon, C.J. and Hedegus, M.C., "Modification and Validation of Conceptual Design Aerodynamic Prediction Method HASC95 With VTXCHN," NASA CR-4712, March 1996.

[6]Heath, C.B., "Wake Vortex Accident Investigations in the 1980's: Four Case Studies," AFWAL-TM-87-179, July 1987.

[7]Kurylowich, G., "A Method for Assessing the Impact of Wake Vortices on USAF Operations," AFFDL-TR-79-3060, July 1979.

[8]Hastings, E.C., Jr. and Keyser, G.L., Jr., Simulator Studies of Vortex Encounters by a Twin-Engine Commercial Jet Transport Airplane," NASA TP-1966, February 1982.

Table 1 Test conditions.

ξ	η	$\zeta(\alpha\ 0)$	$\zeta(\alpha\ 5)$	$\zeta(\alpha\ 10)$
1.50, 2.25	0.0, -0.5, -1.0, -1.5	-0.60	-0.43	-0.27
1.50, 2.25	-0.5, -1.0, -1.5	-0.30	-0.13	0.03
1.50, 2.25	-0.5, -1.0, -1.5	-0.15	0.02	0.18
1.50, 2.25	0.0, -0.5, -1.0, -1.5	0.00	0.17	0.33
1.50, 2.25	0.0, -0.5, -1.0, -1.5	0.60	0.77	0.93

Table 2 Reference values for 10-deg control surface deflections for an F-18 class aircraft.

Control Surface	Coefficient	Value
Aileron	C_l	0.01
Elevator	C_m	0.20
Rudder	C_n	0.01
Rudder	C_y	0.03

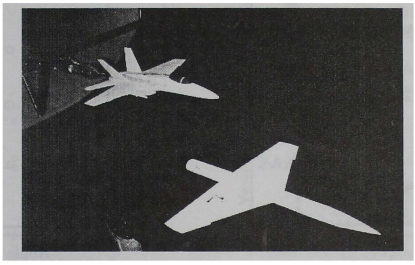


Fig. 1 Aircraft models in 30 ft by 60 ft wind tunnel.

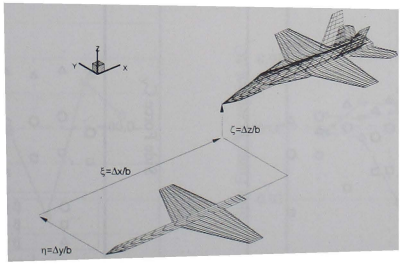
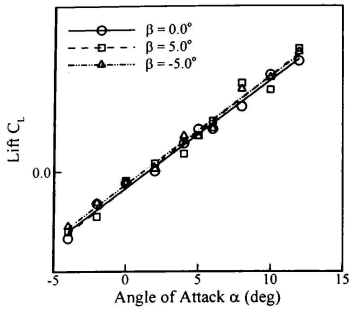
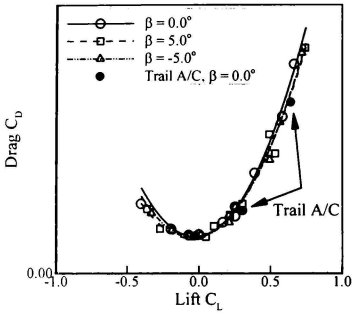


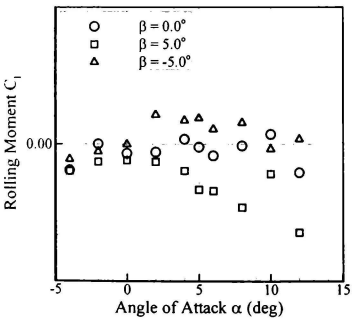
Fig. 2 Coordinate system.



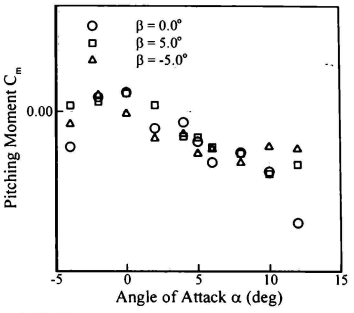
a) Lift.



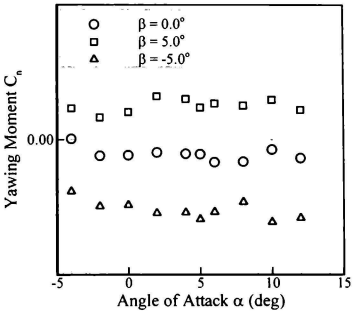
b) Drag.



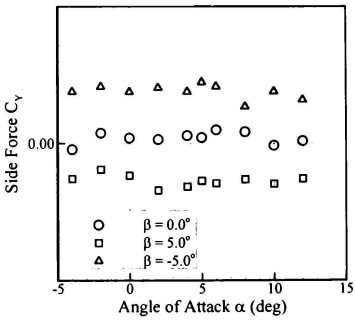
c) Rolling moment.



d) Pitching moment.

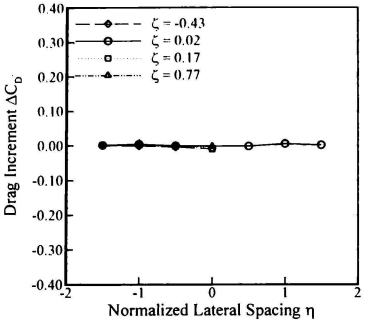


e) Yawing moment.

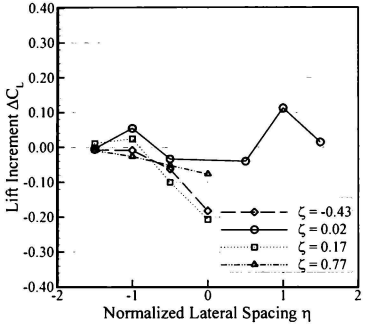


f) Side force.

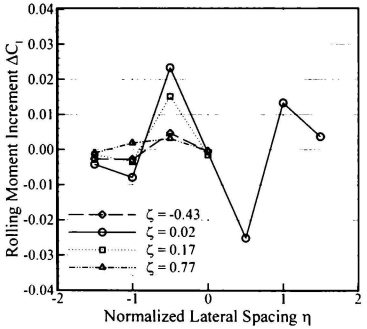
Fig. 3 Baseline data (no lead aircraft) and drag reduction for trail aircraft (b).



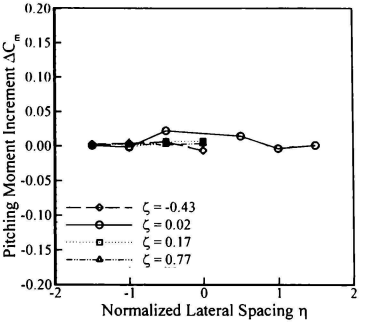
a) Drag.



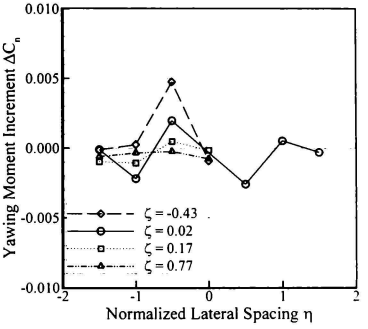
b) Lift.



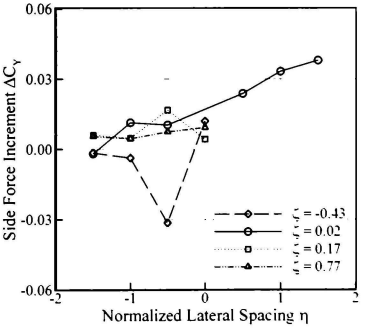
c) Rolling moment.



d) Pitching moment.

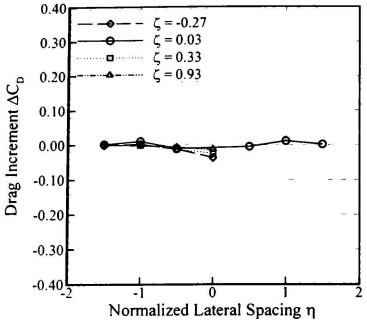


e) Yawing moment.

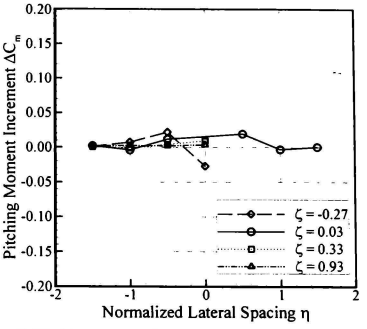


f) Side force.

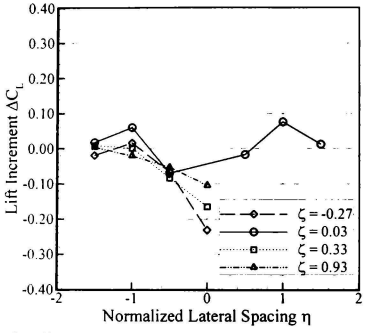
Fig. 4 Position effect on the trail aircraft at α = 5°, β = 0°, ξ = 2.25.



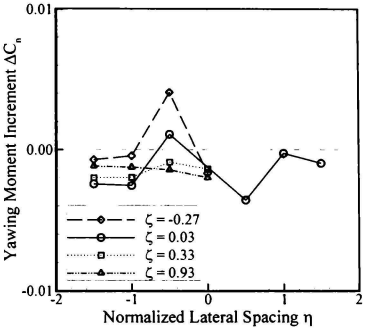
a) Drag.



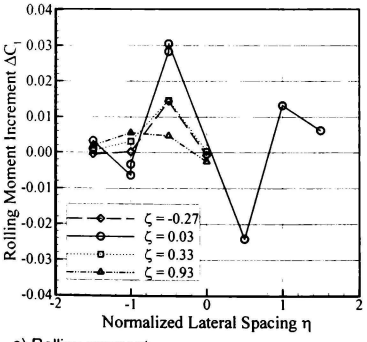
d) Pitching moment.



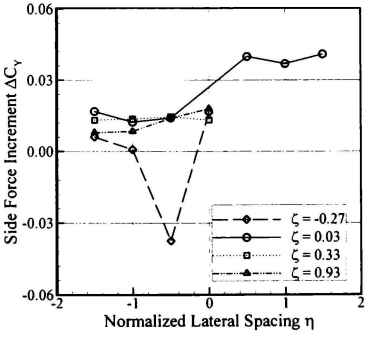
b) Lift.



e) Yawing moment.

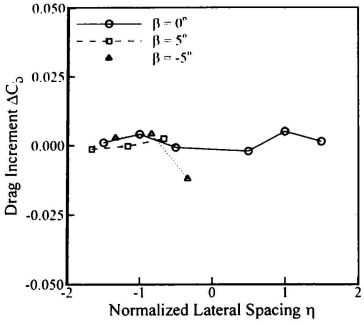


c) Rolling moment.

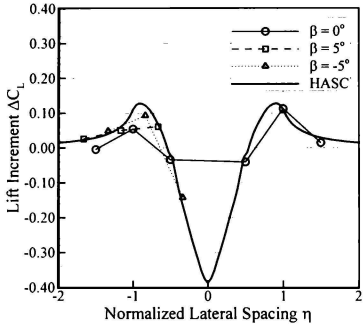


f) Side force.

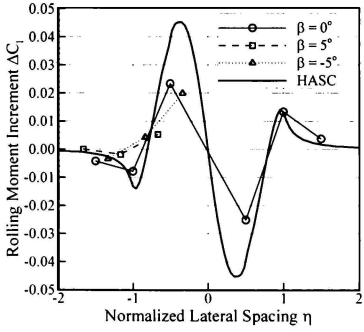
Fig. 5 Position effect on the trail aircraft at α = 10°, β = 0°, ξ = 2.25.



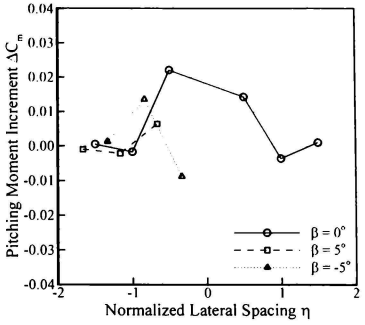
a) Drag.



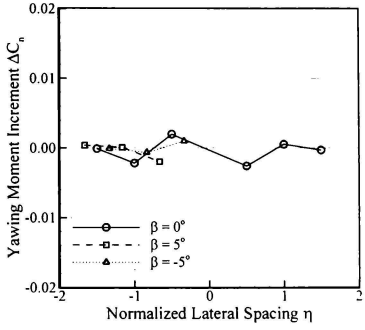
b) Lift.



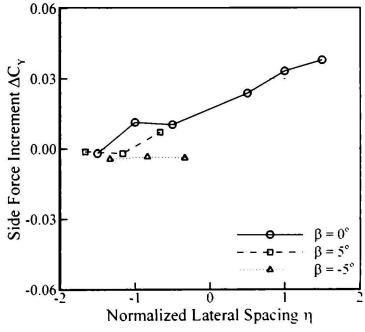
c) Rolling moment.



d) Pitching moment.

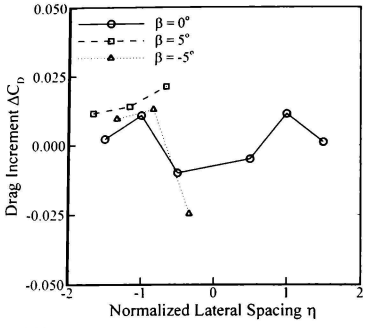


e) Yawing moment.

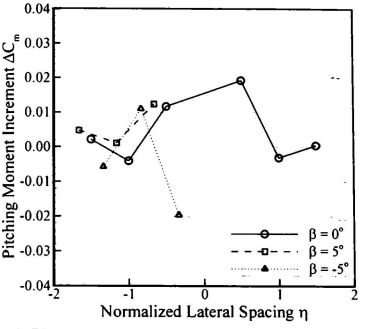


f) Side force.

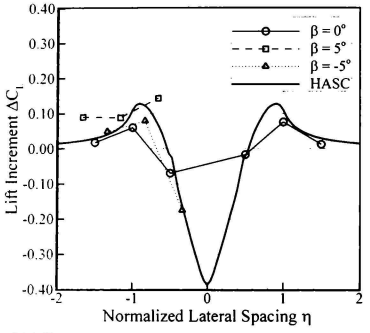
Fig. 6 Sideslip effect on the trail aircraft at α = 5°, ξ = 2.25, ζ = 0.02.



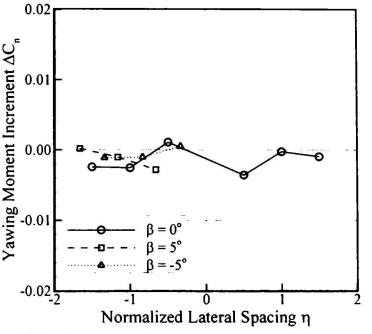
a) Drag.



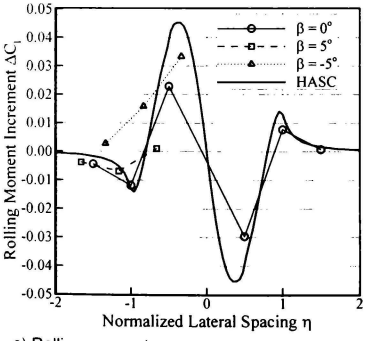
d) Pitching moment.



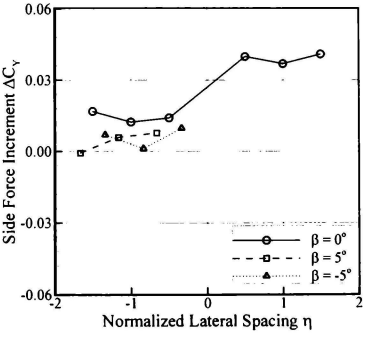
b) Lift.



e) Yawing moment.

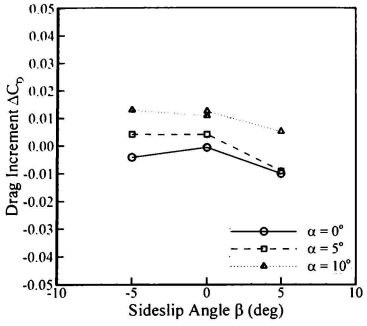


c) Rolling moment.

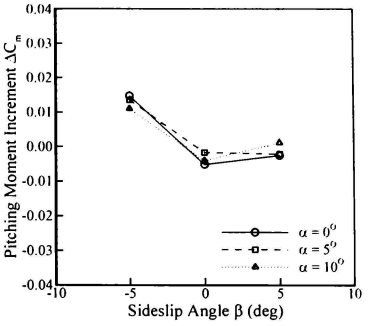


f) Side force.

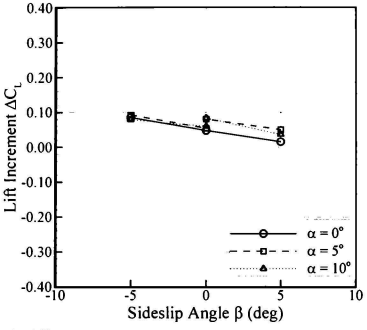
Fig. 7 Sideslip effect on the trail aircraft at α = 10°, ξ = 2.25, ζ = 0.03.



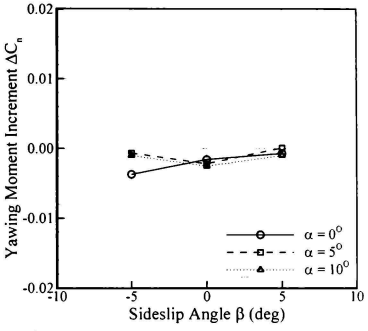
a) Drag.



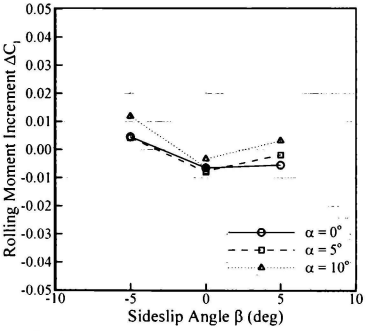
d) Pitching moment.



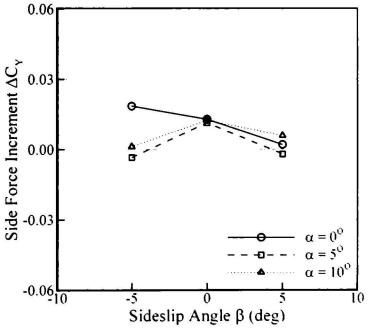
b) Lift.



e) Yawing moment.



c) Rolling moment.



f) Side force.

Fig. 8 Angle of attack effect on the trail aircraft at $\beta = 0^o$, $\xi = 2.25$, $\eta \approx -1.0$, $\zeta \approx 0.0$.

SIMULATION OF AIRCRAFT EMERGENCIES BY MEANS OF DYNAMICALLY SIMILAR MODELS

S. Sadovnychiy, Ph.D., Mexican Petroleum Institute (IMP), Mexico D.F., Mexico. E-mail: ssadovny@www.imp.mx
A. Betin, Ph.D., Scientific &Research Institute for Problems of Physical Modeling, Kharkov, Ukraine.
A. Ryshenko, Dr. Tech. Science, State Aerospace University "KhAI", Kharkov, Ukraine.

ABSTRACT

The complexity of maneuvers in emergency situations does not allow the acquisition of data by means of wind tunnel tests, rocket sleds, rotary machines and other traditional methods of experimental aerodynamics. Statistical data about flight accidents is usually incomplete and may deal mostly with non-catastrophic or pre critical failures. Testing of real aircraft with structural damages and partial system failures is expensive and can expose pilots to high risks. Numerical methods require experimental verification.

On the other hand, the method of flight testing large scaled, free flying, dynamically similar models (DSM) during the development of new aircraft is an efficient way to obtain complete, systematized and operational data about aeroelasticity and dynamics of an aircraft in all types of emergency flight situations.

EXPERIMENTAL PROCEDURE

A free flying model designed for the investigation of flight regimes, is airborne by a piloted carrier (helicopter or aircraft) or by rocket assisted take off (see Fig. 1). The distinctive peculiarity of DSM tests for safety enhancement and survivability is the short duration of the test flight since after partial loss of wings or control surfaces there is no possibility to bring the model back into a working state and repeat the experiment. Thus for such kind of DSM testing, it is enough to have a flight altitude between 600- 1000m and an experiment duration of 5-15 sec.

The most useful data collected is about failures during take-off and landing (for high subsonic speeds of flight V_M=200 to 250 m/s, the model's speed is V_M =50 to 150 m/s). For such low altitudes and speeds it is uneconomic to use piloted aircraft carriers to bring models in adequate

Copyright © 1999 by the
American Institute of Aeronautics and
Astronautics, Inc. All rights reserved.



Fig.1. Model take off by rocket assisted .

experimental conditions, thus, the most effective technique is rocket assisted take off by means of a ramp.

To provide static stability in the take-off regimen the boost stage must have additional control surfaces, which are also utilized for dynamic stabilization in pitch control during launch, where the horizontal stabilizer is ineffective.

After boost stage separation, the DSM conducts all necessary free-flight maneuvers. When the flight program is over, as well as in case of an imminent structural break down of the DSM, the landing system begins the deceleration stage with a drag parachute, it stabilizes the descent attitude, and then decelerates the model by the main canopy.

At the moment of touch down the system stops the descent by means of a complex nose mounted shock absorber avoiding ground wind damage to the model.

Depending on the initial flight regime and the degree of induced structural failure, the model can start unpredictable, non-controllable flight (stall, intensive spin or "across the flow" attitude). Taking this into account, it was necessary for the DSM under discussion to assure a reliable recovery operation. In some cases designers supplied the craft with a specialized emergency system for fast stabilization. On all DSM the deceleration and soft landing system have been fully tested, including parachute and shock absorbers.

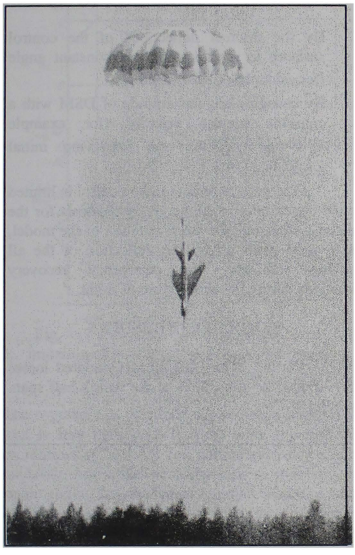


Fig. 2. Model under a parachute

The systems operate as follows. An explosive charge brings into operation the first stage of the parachute system, then, small parachute ($1m^2$ area) stabilizes the model's attitude and begins discoloration to achieve tile adequate speed for deploying the main parachute (Fig.2). The canopy of main parachute has a 60 m^2 area, and provides deceleration up to a vertical speed as 7 m/s. The landing occurs nose down on the mechanical shock absorber which penetrates the soft ground and cancels vertical and horizontal speed. With a wind velocity up to 4 m/s the model remains in a vertical position, protecting its wings and other protruding surfaces from ground damage. After flight data processing, carrying out technical maintenance and readjustment of onboard systems for a new flight program, the model is ready to fulfill further tests.

METHODOLOGY

In all experiments the type and degree of failures were strictly determined in concordance with adjustments of onboard systems for failure simulation on the DSM, such as size and shape for the detaching components of a wing and control surfaces, and decreasing the rigidity of servo drives. This approach allows to fulfill a series of flight tests and define critical maneuvers for simulated type and degree of a failure. Also it allows to check data recurrence and comparing it with other test methods.

The predetermined variables for various types of failures are the following:

a) the type of maneuver, realized at the moment of failure simulation;
b) the performance demand of a maneuver at a given moment (for example, overload factor);
c) the position of the model's center of gravity;
d) velocity head t the experimental regime;
e) autopilot's control laws.

EXPERIMENTAL SUBJECT

As a subject for designing a series of flight tests we consider the model, which is an aircraft of a classic aerodynamic scheme (Fig.3). This aircraft has highly swept wings a moderate wing elongation and all-moving horizontal tail surfaces.

DSM data:

fuselage length	3.36 m
wing span	1.784 m
mean aerodynamic chord (MAC)	0.803 m
wing sweep angle	63°
mass	156 kg
weight of onboard equipment	45.6 kg
weight of the rescue system	17.8 kg
test velocities	50... 120 m/s
test altitudes	300-1000 m
take off weight	223.55 kg
take-off thrust-to weight ratio	3.6

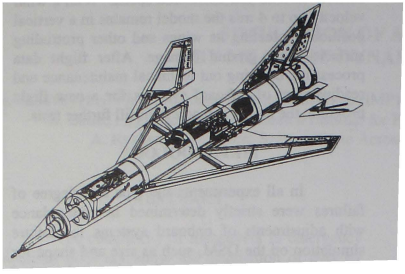


Fig. 3. Structurally-aerodynamic scheme of model.

The main conditions for an adequate representation of an aircraft by the model, concerning damaged load carrying and control surfaces, or onboard equipment failures, require the following similarities, before and after damages are induced: geometry, cinematic and dynamic, meaning similarity of Newton, Froude and Strouhal numbers. During the investigation of aeroelasticity phenomena it is also necessary to provide similarity of Cauchy criteria.

The model and the aircraft have the same control surfaces (ailerons, rudder, and all moving tail); the deflection angle range is also equal. The engine's air passage and thrust have not been modeled, and a conical nose fairing was used instead of the air intake.

Geometrically similar surfaces of the fuselage and wings of the model are made of load bearing sandwich panels.

The bearing plies of the panels are produced from composite materials based on fiberglass and carbon fiber tape with epoxic binder and contain a plastic aggregate.

At the points of concentrated stresses load bearing metal elements are installed. Airfoils are usually thin, (i.e.: the detachable part of the wing, vertical stabilizer, control surfaces), so they are the most exposed to failures during tests. For this a more practical structural design based on composite skin and compact plastic foam has been utilized.

Onboard equipment includes the following systems: control, flight data measurement and registration, electric power, deceleration and soft landing.

The automatic control system provides the flight program on a preset trajectory by means of the deflection of control surfaces, according to the following laws:

$$\delta_e = G_\theta(\theta - \theta_t) + G_{\omega_y}(\omega_y - \omega_{y_t}) + \delta_t$$

$$\delta_a = G_\gamma(\gamma - \gamma_t) + G_{\omega x}\omega_x$$

$$\delta_R = G_\psi(\psi - \psi_t)$$

Were : δ - surface deflection (elevator, aileron, rudder); θ,γ and ψ- pitch, roll and yaw angles, ω - rate of angles, G - gain of autopilot, subscrites t - task value of parameters. G_θ=0.85, G_{ω_y}=1.13, G_γ = 1.1, G_ψ =0.55.

There are two ways of realizing the maneuvers in the experimental path of the flight:

1. By the discreet deflections of the control surfaces to a pre-established constant angle (for example $\delta_e = \delta_t = const$).
2. By changing angular attitude of DSM with a constant angular velocity (for example $\theta_t = \theta_{t0} + K_\theta * t$) and preserving initial control laws.

The ground remote radio control is limited to the operation of the emergency channels for the rescue system, and for the orientation of the model, as in deep pitch up by the deflection of the all moving stabilizer, for emergency recovery situations during the active stage of flight.

FLIGHT TEST RESULTS

During of investigations simulated losses of: 20%, 28%, 40%, 50% of the wing's half span. (see Fig.4).

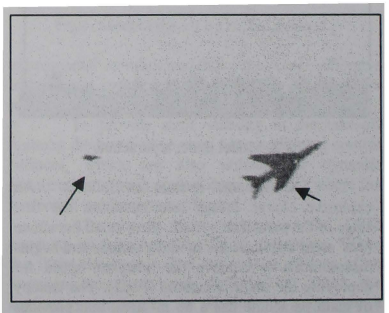


Fig.4. Moment of the loss of the wing's half span

It has been determined that the loss of 28% (see Fig.5), in strait and level flight, the automatic flight control system (using the rest of the undamaged control units) is able to counteract short term disturbances that appear in the model.

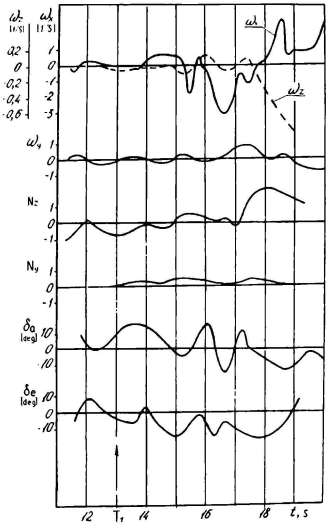


Fig. 5. Parameters of model flight with imitation of loss 28 % of a wing's half span. V = 90 m/s; H = 600 m.

The overshoot of angular roll speed ω_x, during this loss is less than 0.35 Rad/sec with a period of oscillations of 0.4 sec (all parameters refer to the model). After, the model passes to a relatively stable flight: the change of angular velocities and overloads have an aperiodic character with small absolute magnitudes, just slightly above usual perturbances of the model's free flight that result from casual factors. The flight of an asymmetric aerodynamic scheme differs from the symmetric in small constant components of lateral overload $N_y \approx 0{,}25$ and aileron angle $\delta_a \approx 8°$
This angle corresponds to the aerodynamic balance for such asymmetric scheme.

With a 40% loss, the disturbances were much more intensive (see Fig.6) immediately after failure induction, the oscillations ω_x, reach 2.6 ... 3.1 Rad/Sec with the same periodic component. The flight of the model occurs with a build-up of oscillations and with increasing roll.

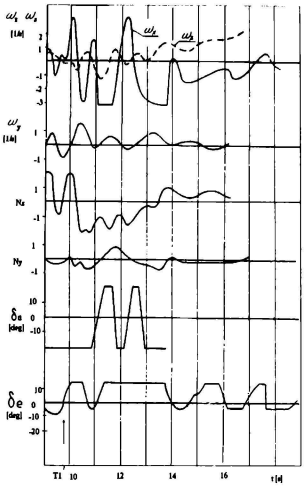


Fig. 6. Parameters of model flight with imitation of loss 40 % of a wing's half span. V = 65 m/s; H = 550 m.

The angular velocity has also an oscillatory pattern of change, but during first three seconds after the loss has happened, the constant component is still absent. The more intensive processes in this flight are connected not only with the larger area lost off the wing, but also with large overloads appearing after the failure occurred ($1 \leq N_y \leq -2.2$). The model enters a spin (overload N_y changes and the constant components of angular velocities increase) just in 2.7 seconds after failure started and by one second before the cancellation of the maximum deflection of the all-moving stabilizer needed for a pitch maneuver.

Obtained data shows that for the model of the aerodynamic scheme under consideration the critical loss of wing area for flights with speeds V_m=60 to 70 m/s and overloads N_y = -1,5 is 40% of half span.

With a loss of 50% (see Fig. 7) there is no controllable flight because the change of angular

roll velocity exceeds the effective range of a sensor 3.14 *Rad/sec* even under small overload factor ranges ($N_z \approx 0.5$) in 0.8 to 1 *sec* after the start of failure simulation; at speeds of 70 m/s the model transitions into intensive left rolls.

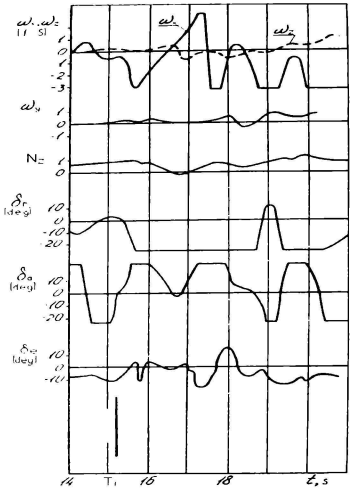


Fig.7. Parameters of model flight with imitation of loss 50 % of a wing's half span. V = 90 m/s; H = 600 m.

During flight tests of DSM it was shown that a flight with an overload of $/N_z/ = 1.5$, at speeds with maximum controllability, such kinds of failure as the loss of 40% of wing with its aileron is close to critical; during which it is impossible to fulfill a flight task or even to return to the airdrome or to an alternate airfield.

For increasing the aerodynamic survivability of an aircraft it is necessary to develop a reconfigurable flight control system, With 28% loss of the wing's half-span we may state that the real aircraft is able to continue controlled flight.

As a second characteristic example of a DSM flight test, we may consider the results of investigating the failures leading to a sharp drop in stiffness of the control units in the pitch channel (with the preservation of friction couplings, installed on the shafts of an all moving stabilizer).

A special attachment for stabilizer's surfaces with elastic similarity was developed for the investigation of the aeroelastic processes caused by feathering of the all-moving stabilizer during severe pitch maneuvers. This attachment is installed on the sides of the fuselage by means of arms and beams equipped with self aligning spherical bearings. The stabilizer's shafts pass through these bearings and have inside the fuselage a rectangular cross-section that act as elastic elements, providing a given bending rigidity in the chord's plane. Close to the spherical bearing on the shaft a second elastic element is attached. It's deformations provide torsional rigidity of the control surface. This elastic element through a control rod and crank is attached to the servo drive of the all moving stabilizer. Moreover, a strong elastic element with a "V" shape is attached to the end of the shaft. The deformations of this last element provide rigidity of the surface attachment in the plane perpendicular to the chord. Thus, the system of flexible elements provides a given rigidity of the stabilizer's surfaces and allows to investigate special form of "flutter on maneuver" that is characterized by intensive oscillations of the all moving stabilizer in all its degrees of freedom (including oscillations in the chord plane). This kind of flutter has also interdependence between the loss of aeroelastic stability and the aircraft's maneuvers. Mechanical stoppers of oscillation amplitude, as well as a system of emergency flutter suppression averts the danger of excessive deformations, that may break the airframe. DSM flight test allowed to investigate flight stability up to a speed of 130 m/s with the simulation of damages in the pitch control channel unit. It also allowed to investigate the stability of elastic oscillations of the stabilizer with a pre established rigidity of its joint.

It was determined that during a simulation two aeroelastic processes take place:

a) low frequency phenomena, and

b) high frequency oscillations

Low frequency process represents oscillations of both surfaces of an all moving horizontal stabilizer, around a position given by the flight control system. The deflections of the stabilizer are tied with severe pitch maneuvers.

An example of flight data is presented on Fig. 8. The rigidity of the control drive was diminished by 17.3 times. The flight experiment was initiated at a velocity head of 8,830 N/m^2 (velocity of the model is 120 m/s). The cause for the start of low frequency oscillations was in the expected trajectory disturbance overload, that augmented from 0 to +2 during 0.7s. The lift of the stabilizer increased respectively (maintained by servo-drives in a trim position). The hinge axis of

the stabilizer is situated between subsonic and supersonic positions of the model's aerodynamic.

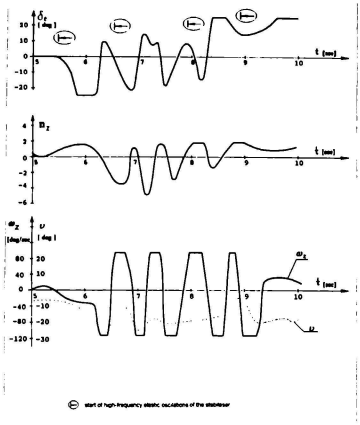


Fig. 8. Low frequency process

At subsonic speed an increment in stabilizer lift causes additional torque around the hinge that causes further deflection, in turn producing more lift. After simulation of servo drive rigidity loss the servo drives could not prevent this tendency. Stabilizer was deflected almost up to its limit. Only during the models maneuver (sharp change of angular velocity ω_y, from 15 deg/s to 100 deg/s and, most important, a change of load factor N_z from + 2.1 to -2 that diminished the lift of both surfaces) servo drives of low rigidity could deflect the stabilizer initial position. The process of stabilizer reversal from one limit to the other continued up to the end test flight .

The second high frequency process of elastic oscillations was registered in flight tests at certain phases of the low frequency process. The typical episode of the beginning of such kind of oscillations is presented on Fig. 9. At 7.25 s of flight the stabilizer with diminished drive rigidity was deflected by airflow to an angle of 22.5° during pitching.

Hereto elastic oscillations due to flutter appeared. The oscillatory character of the bending moment acting at the stabilizer plane chords M, as well as for torque M_T and bending moment M_P acting perpendicular to the chord's plane, is clearly seen in Fig. 9. The oscillation frequencies of the stabilizer are close to the frequencies found during ground tests.

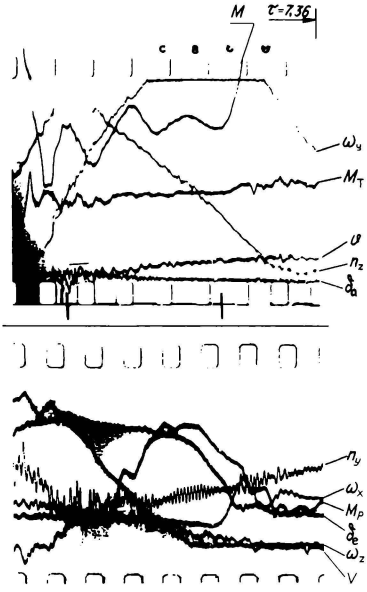


Fig. 9. High frequency process of elastic oscillations

The cause of stabilizer's flutter is the energetic maneuver of the model, which in turn was caused by the detection of feathering stabilizer at the maximum angle (22.5 degrees). Angular velocity of this maneuver changes from 130 to 140 deg/s, and the load factor change was from -6.6 up to +2 (see Fig. 6). Obtained data allow us to state when the modeled failure is super critical for an aircraft and it is impossible to continue flight.

CONCLUSIONS

These tests and investigations have shown the advantages of researching using DSM, for adequately represent aeroelastic processes in interaction with the most complex aircraft maneuvers.

REFERENCES

1. Holleman E.C. Summary of flight tests to determinate the spin and controllability characteristics of a remotely piloted, large-scale (3/8) fighter airplane model. - NASA TN D-8052, 1976. 124 p.
2. Michael A. Dornheim. X-36 To Test Agility of Tailless Design. - Aviation Week & Space Technology, March 4, 1996, p. 20-21.
3. Patuxen River. NASA Drop Model Sinks. - Aviation Week & Space Technology, January 20, 1997. P.59.
4. S. Sadovnitchii, V. Ryabkov, A. Rushenko, J.Sandoval. "Modelling of Aircraft Flight by Means of Dynamically Similar Models with Flight Control System Similarity". Proceed. of American Institute of Aeronavtics and Astronavtics, Modelling and Simulation Technology Conference-97, pp.326-332., New Orleans, U.S.A., 1997.
5. S. Sadovnychiy, A. Betin, A. Ryshenko, R. Peralta. "Simulation of Aircraft Flight Dynamics by Means of Dynamically Similar Models". Proceed. of American Institute of Aeronavtics and Astronavtics, Modelling and Simulation Technology Conference-98, pp.64-69, Boston , U.S.A., August 10-12, 1998.
6. K. Cunningham, P. Keney, D. Leslie, D. Geyer, M. Madden, and P. Glaab "Simulation of a F-18 E/F Drop Model Using LaSRS++ Framework". Proceed. of American Institute of Aeronavtics and Astronavtics, Modelling and Simulation Technology Conference-98, pp.9-16, Boston , U.S.A., August 10-12, 1998.

AIAA-99-4196

MULTI-PURPOSE DOMED SIMULATOR

William R. Edmiston and John A. VanHoogstrate, Sr.
The Boeing Co.
St. Louis, Mo.

Abstract

In 1998 it was determined that in addition to the X-32/Joint Strike Fighter (JSF) developmental flight simulator(s) in Seattle, a similar capability was required in St. Louis. Creation of a new simulator and an accommodating facility for this purpose was ruled out due to cost. Instead it was decided to modify an existing pilot-in-the-loop simulator and associated facility. The modifications had to be affordable and the particular facility and simulator would have to serve as a multi-purpose test article since the assets of the St. Louis Flight Simulation facilities are deemed to be "shared" company resources. As a result, an existing fix-based, domed simulator was chosen for modification with the multi-purpose, multi-use objectives in mind. The resulting modifications and enhancements to the candidate simulator and facility have met the requirements of the X-32/JSF control law development program as well as other existing programs. Economic benefit was derived since the cost was less than one tenth that to design and build a new simulator.

Introduction

The Boeing Company Flight Simulation Facilities located in St. Louis, Missouri, see Figure 1, have a long history of supporting the development of tactical fighter aircraft for the USAF, USN and USMC.

Each 40 foot diameter dome houses a fix-based simulator and is located in an individual, secure physical module, commonly referred to as a Manned Air Combat Simulator or MACS. In addition to the crew station and dome systems, each MACS includes a test operator station as well as computer, image generation and data recording systems.

Copyright © 1999 by the Boeing Company. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

Each domed simulator is capable of operating in a "stand-alone" mode or linked to other manned simulators via secure local and/or wide-area networks.

In 1998 the Boeing JSF program management decided a piloted X-32-like simulator was required in St. Louis to support flight control law development. Early on it was deemed that financial constraints made it impossible to build a new simulator replicating the X-32 geometry and cockpit display configuration. As a result, an alternative approach was to be found. However, the one requirement that could not be compromised was to provide projected out-the-window scenery that would be suitable for performing vertical flight maneuvers and carrier approaches and landings. So, basically, what was required was to identify an existing resource which could be modified at reasonable expense and retain its usefulness to other programs.

Of the options available, it was decided that the most suitable candidate was the Advanced Programs module, MACS 6. At the time this facility included a standard three channel, out-the-window projection system projecting onto the dome. Installed within the dome was a "reconfigurable", generic crew station equipped with a McFadden hydraulic control loader system, see Figure 2. The original and primary purpose of this simulator was to support the company's advanced programs, particularly those developing and evaluating aircraft flight control systems and handling quality criteria. Over time, the simulator became a general-purpose test article supporting the T-45 program and a number of internal R&D programs. Because the single-seat crew station was configured with a large CRT and McFadden system, the cockpit displays and controls were programmable; consequently, many kinds of military tactical aircraft (friend and foe, alike) have been "flown" using this simulator. This facility and crew station seemed to be the logical choice for performing X-32/JSF simulations.

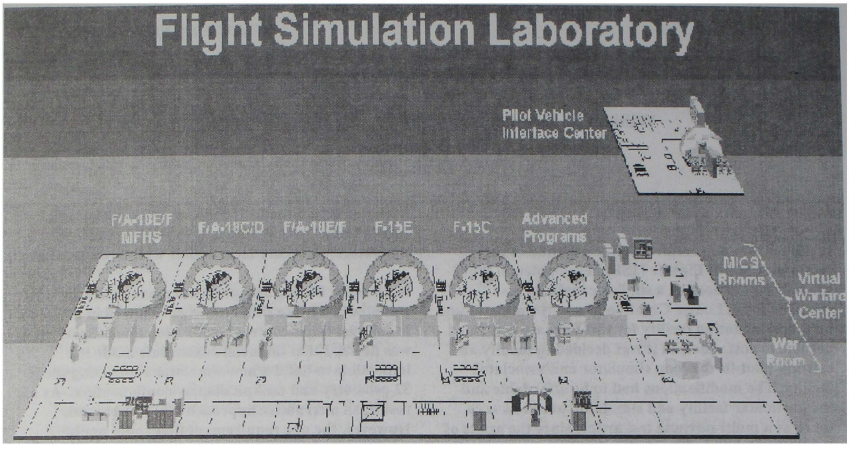


Figure 1: Boeing Flight Simulation Facility - St. Louis

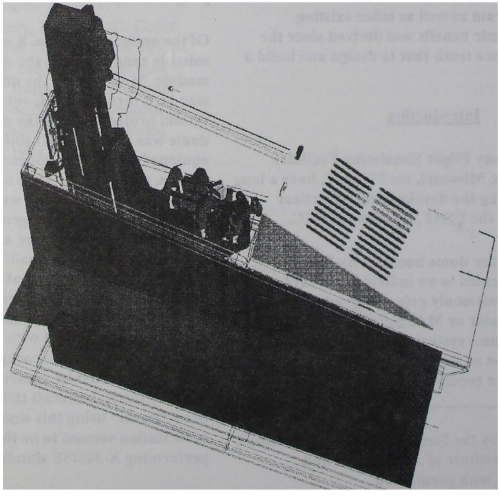


Figure 2: MACS 6 Crew Station

Requirements

When this simulator was proposed for X-32 program use, it was decided that it had to be suitable for not only control law development and flying/handling quality evaluation but for supporting the flight test program, as well. As a result, the top-level requirements included the following:

The fidelity of the out-the-window visual scenery needed to be suitable for vertical, carrier and conventional airfield operations
The geometry of the crew station needed to be representative of the X-32, (e.g. the over-the-nose visibility)

- The flight controllers needed to be representative of the X-32, (e.g. stick, pedals and throttles)
- A basic X-32 heads-down display was necessary
- A visual data base for the USN test facility at Patuxent River was necessary to support subsequent flight testing, (the Edward AFB airfield already existed)
- Overall simulation transport delays needed to be minimized

The proper HUD field-of-view (FOV) and content needed to be portrayed
Software commonality between Seattle and St. Louis needed to be maintained

More specifically it was decided that the simulator crew station needed to have the proper over-the-nose visibility, the correct seating arrangement and the proper flight controller configuration, i.e. stick, throttles and pedals. The existing McFadden system was suitable to provide the required force-feel for the stick but a new throttle with the appropriate mechanization and functionality was required. Although it was desirable to replicate the Boeing - Seattle X-32 simulator, this was not feasible from the standpoint of cost. The most economical approach was to modify and complement the existing crew station and dome systems in order to fulfill these requirements.

The multi-purpose criteria dictated that high resolution out-the-window (OTW) visuals were required for assessing air-to-air, air-to-ground, carrier and vertical flight operations for a variety of tactical and trainer military aircraft. As a result, it was determined that a multiple projector system capable of providing 2 to 6 channels of OTW video in a variety of arrangements was required. The reason for this was twofold: first, programs using the facility require different levels of fidelity, coverage and resolution and, second, the number of channels of image generation video that are available varies on a daily basis.

Approach

As stated, the MACS 6 facility required an enhanced out-the-window projection system. Based on previous AV-8B and helicopter simulation experience the desirable OTW scene for such a task called for 140 to 180 degrees of high-resolution imagery in the forward FOV. In order to achieve this a multiple number of IG channels would be required to provide the necessary resolution and 180 degrees FOV. The MACS 6 dome was originally supplied with only two forward FOV projectors. It was surmised that four projectors would be required for the forward FOV.

The over-the-nose visibility of the existing crew station did not meet the required 17 degrees for the X-32. The most expedient way of rectifying this was either to reposition the existing CRT in the crew station or replace it with a low profile, high definition monitor. Repositioning the existing CRT meant modifying the floor of the crew station in order to accommodate the pilot. It was decided to replace the existing CRT with the lower profile monitor.

Another critical factor was duplicating the "hands-on throttles and stick" (HOTAS) controls of the X-32. The most expedient and affordable approach here seemed to be to design and install physical and electrical adapters on the stick column for the purpose of accommodating the X-32 flight controller grip. In that way rapid changeovers were possible. The throttle, however, was a more complex situation. The existing throttle quadrant in the MACS 6 simulator did not have the required functionality of the X-32 throttle quadrant. It was apparent that a new simulated X-32 throttle quadrant was required. However, it was also planned that the two throttle quadrants could be physically exchanged in an expedient manner.

The Heads-Up Display (HUD) is projected onto the MACS 6 dome surface. The reason for this is to minimize focus and parallax problems encountered when using a flight hardware HUD. The projector used for this purpose was a somewhat antiquated system that could only provide a "standard" narrow field-of-view type of HUD display. A new projector with a wider FOV was required for the X-32 HUD along with a new mounting arrangement in order to produce a non-distorted image.

The final consideration involved assessing the required "compute" power for the X-32 simulation. Prior to the introduction of the X-32 simulation into MACS 6, a Dec AlphaServer 2100 host system was being used. The requirements for the X-32 "real-time" simulation were somewhat unknown since the Boeing X-32 engineering development team was providing the aircraft software models and control laws. As the result of the initial simulation timing and performance assessment, it was quickly decided that the 2100 system needed to be replaced with something more powerful.

MACS 6 Projection Systems

The need for a realistic, high fidelity out-the-window visual display in the MACS 6 dome facility prompted an upgrade of the facilities projectors with state-of-the-art systems. Changes to the MACS 6 projection system were made possible due in part to new commercial-off-the-shelf projection technology and the availability of products that fitted our needs. A convertible or reconfigurable capability was required in order to provide a variety of display configurations for serving multiple users.

User requirements ranged from 360 degrees coverage to 160 to 180 degrees of high fidelity coverage in the forward FOV. The total number of OTW imagery projectors in the MACS 6 dome eventually numbered five.

The two forward FOV projectors in MACS 6 were replaced and two more added. The external-to-the dome philosophy for projector mounting was retained. Dual projection platforms, 30 feet high, hanging over the outside of the dome's structure were constructed, see Figures 3 and 4. Stairs and a catwalk, all of which are supported by ceiling and wall structure, connect the platforms. Care in the placement of these platforms and catwalk was taken to maintain access to an existing lower projection platform. The two added projectors are located 60 degrees left and right from the back of the dome and 45 degrees above dome center. This position allows projection across the dome clearing the top of the pilot and cockpit. The installation of the outboard, inset projections is designed to provide a continuous, high fidelity panoramic display and to add image content to the low, off-nose area. The resulting 4-window configuration for the X-32 simulation provides the necessary forward FOV coverage, see Figure 5.

Each projected image is corrected for geometrical distortion via software hosted on the CompuScene image generator system. All mating edges are butted, eliminating the overlapped visual effect.

Hemispherical Projector

Recent commercial product lines allowed for replacement of the MACS 6 Talaria wide FOV projector and its complex hemispherical projection lens assembly. The Barco Reality 9200 LCD Projector was chosen as our new hemi-projector, primarily for its wide angle High Definition (HD) zoom lens. Mounted on-axis with the LCD light source, this lens produces a 180 degrees circular field whose diameter is the height of the LCD raster. The orientation of the forward hemi projector provides a vertical coverage of 63 degrees forward and above the design eye and 180 degrees in the horizontal. The aft hemi projector fills in the rest of the dome. This matches the coverage provided by the previous system. This Barco projector is rated at 4,000 ANSI Lumens, a considerable step up from the 1000 Lumen Talaria projectors previously used. At dome operational settings, the new 9200 projector produces a comfortable viewing image whose highlight brightness is approximately, but not limited to 0.7 Foot Lamberts. The painted, matte white dome surface is 85% reflective due to integrating effects. This results in a 6:1 contrast ratio. The resolution of the new projector with the CompuScene image generator has not changed from the old system, (a 4:3 aspect of 960 pixels resulting in an angular subtend of 11 ¼ arc minutes per pixel) but picture quality has. We are now operating at 4x the brightness and 2.5x more contrast than previously.

The Barco Reality 9200 projector is equipped with "programmable blanking" via software hosted on a laptop computer. This option provides the capability to create blanking masks on a pixel-by-pixel basis. As a result this capability can be used to eliminate unwanted reflections and other interference in the dome as well as provide the proper cutout for inset projection.

Inset Projectors

The projector system of choice to replace the Talaria inset projector was the Barco Reality 8200 LCD projector. This high-resolution projector supports up to 2,000 x 1,280 pixels and will produce nearly 2,000 ANSI Lumens. It was selected over other manufacturer models due to its performance,

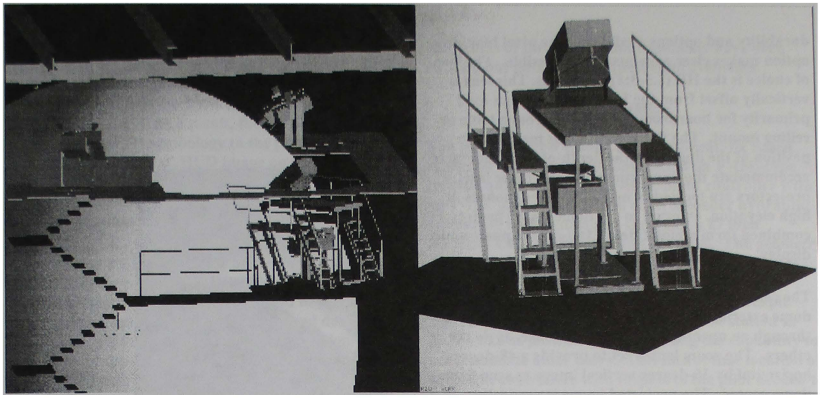


Figure 3: Projector Installation

Figure 4: Forward Inset and Hemi Projector Platforms

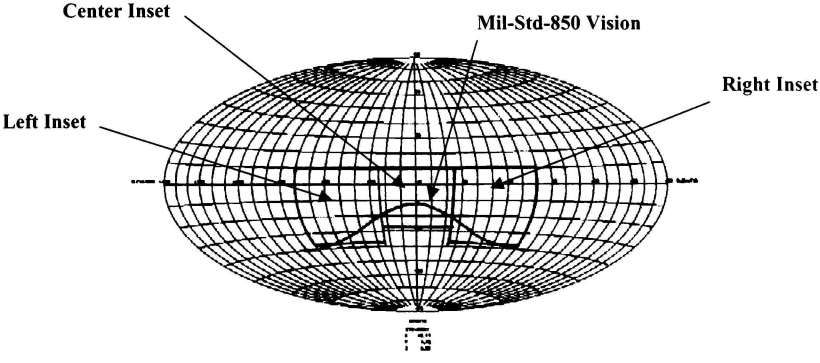


Figure 5: MACS 6 Projector Coverage

durability and options available. The pixel blanking option makes close edge matching possible. The lens of choice is the HD (1.5-3:1) zoom lens. This lens is vertically offset from the LCD axis: a design used primarily for boardroom projection from a table or ceiling mount. The flat field of focus matches the position of the dome and the depth-of-field enough to accommodate the dome curvature. All three inset projectors are located in newly designed mounts at high elevation, external to the dome. All 3 projectors combined form a high fidelity 160-degree panoramic display.

The center inset projector is mounted outside the dome exterior, above and behind the pilot. It projects through an opening on the dome surface, as do the others. The zoom lens is set to provide a 48-degree horizontal by 36-degree vertical image as seen from dome center. The projected image is vertically positioned to cover an area of 9 degrees above the design eye and 27 degrees down. Angular resolution from the design eye is 2 ¼ arc minutes per pixel. At operational settings, highlight brightness is 2.8 Foot-Lamberts and the contrast ratio is 7:1, a 2x increase in contrast from the previous system. We no longer concern ourselves with 'what level brightness?' but simply set it to a comfortable viewing level.

The two side projectors have the same lens type as the center projector. They are set to a shorter focal length, projecting a larger image: 56-degrees horizontal by 47-degrees vertical. Each projector is positioned to provide vertical coverage of 9 degrees above design eye and 38 degrees below. Angular resolution is 3 arc minutes per pixel. At operational settings, the two side projectors nearly match the center projector with a highlight brightness of 2.4 Foot-Lamberts and a Contrast Ratio of 7:1. The bright pupils of the two side projectors have not been a problem in the 40 foot diameter dome of MACS 6. If warranted, the lens opening of a projector can be shuttered based on where the area of interest is at the time. For example a head-tracking device could be used to shutter a lens.

All inset projector mounts have been designed to rotate left /right and up/down. In addition to side-by-side image placement, images can be vertically stacked to accommodate air-to-air refueling simulations. An inset within an inset is also possible. The zoom lens can reduce images to 36 degrees horizontal x 27 degrees vertical resulting in a resolution of 1.67 minutes per pixel. Such a setting can enhance the scene quality for carrier approach and landing scenarios. Changes in image placement do require the appropriate distortion correction algorithms and calibration software. All these variations provide the flexibility to produce a significant number of out-the-window configurations based on need or objectives, see Figures 6, 7 and 8.

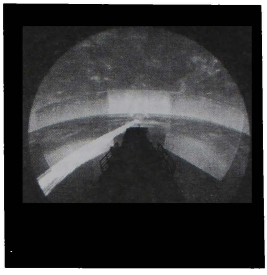


Figure 6: X-32 with 4 Windows

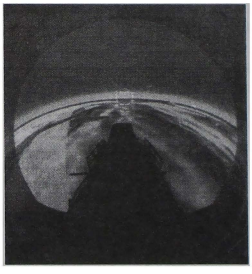


Figure 7: F-15 with 3 Windows



Figure 8: T-45 with 2 Windows

Heads-Up Display Projector

The use of a flight-hardware Heads-Up-Display (HUD) with domed simulators results in parallax variations between the perceived HUD image and the OTW projected imagery. As a result, the preferred method of presenting HUD symbology to the pilot is to project or superimpose a 'real' HUD image on the dome surface. A Silicon Graphic IG system is used to produce the symbology and a CRT projector displays this onto the dome surface.

By mounting a CRT projector below and forward of the cockpit, the projected HUD symbology is reflected from a folding mirror (necessary due to the limited space in this area) upward onto the curved dome surface, see Figure 9. The distortion correction capability of the projector eliminates keystone and other curvilinear errors. Only the GREEN, center lens/CRT is used for this application since the HUD is monochromatic

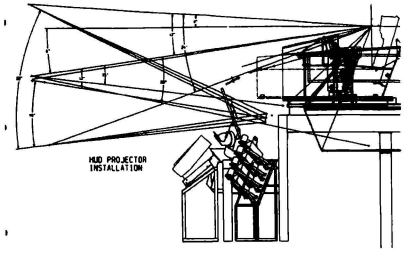


Figure 9: HUD Projector Installation

The vintage Electrohome projector being used to display HUD symbology on the dome surface of MACS 6 could not produce the required FOV for the X-32 simulation. In addition to this, it produced a less-than-desired image quality. Pilots complained that the HUD was difficult to read. After assessing these shortcomings and user requirements, an Electrohome model Marquee 8110 (CRT) projector was chosen as a replacement. Any size or shaped HUD image up to 34hx26v degrees can now be produced. This means that the depression angle and FOV requirements of all known users at this time can be accommodated. There are calibration procedures for adjusting HUD position and FOV. The 'reading' quality of the HUD has also been remarkably improved and controls were added to adjust the brightness of the HUD. Pilot comments have been nothing but positive since these improvements.

Crew Station

The MACS 6 crew station is a single-seat, fix based simulator and originally consisted of an F-15E-type throttle quadrant, a center-stick flight controller with an F-15E grip, a McFadden control-loader hydraulic system for both pedals and stick and a 27 inch DataRay CRT for heads-down displays. The geometry of the crew station was generic in the sense that it was designed to accommodate the original equipment rather than to duplicate an actual aircraft cockpit. The challenge then was to make the crew station as X-32 - like as possible while retaining the capability to accommodate other users. This meant that location of the flight controllers with respect to the pilot needed to be as correct as possible, the over-the-nose visibility had to be correct and the heads-down displays needed to be satisfactory. In addition, an active throttle quadrant was required.

In order to satisfy the 17 degrees over-the-nose visibility of the X-32, a low profile Panasonic (16:9) 28 inch wide display monitor was chosen to replace the original 27 inch CRT. The electronics of this monitor were then "repackaged" for installation. A new, low-cost touch screen device was also needed as a result. An infrared touch sensor was designed and built to fit the Panasonic monitor by Advanced Touch Applications; this same firm had previously built similar units for Boeing's F/A-18E/F engineering simulators.

The throttle quadrant that was designed and built in-house was made to be interchangeable with the original quadrant. In addition, an adapter was made to fit the flight control stick column in order to accommodate the X-32 grip. This adapter can be readily removed and other grips installed as necessary. As a result, changeover to other flight controller configurations is only a matter of minutes. All the controllers are located within .1 to .2 inches of X-32 design specs (in all axes), Figure 10. The final touch to the crew station was to build and install a seat adjustment bracket that allows for the seat tilt-angle to match that of the X-32.

Dome Systems

One of the primary concerns of any real-time, piloted simulation is that the overall (computer, software and hardware) system delays be minimized. In the case of the X-32 simulation, it was deemed that a faster host

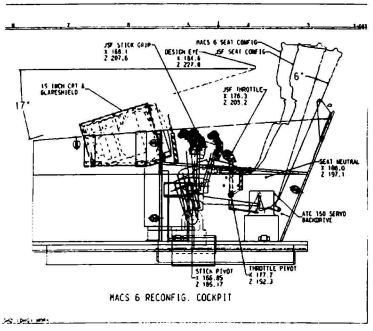


Figure 10: X-32 Simulator Configuration

computer would be required in order to process prototype operational flight software at the appropriate computational rate. The DEC (now Compaq) 4100 was chosen as the new host computer. There were several reasons for this: price, performance and commonality with other Boeing simulation facilities. This computer has four Alpha CPUs, each with 533MHz performance. Thus far, this host system has proven to be more than satisfactory. The overall time delays of the simulator are an average of 105 msec to the end of the CompuScene IG system's first video frame. More importantly the 4100 can match the sampling, duty cycle and output rate of the X-32's integrated flight-propulsion computer (IFPC).

The other facility computer system that required an upgrade was the IG system used for creating heads-down displays. This older SGI system was replaced with a SGI Octane capable of producing line rates of 1600x1200 for use with the Panasonic high definition monitor. Other ancillary computer and IG computer systems were deemed acceptable.

Summary

The MACS 6 systems and crew station have been modified and enhanced for the purpose of supporting the X-32 control law design task and for evaluating predicted handling and flying qualities. Excluding the OTW projection systems the cost of this upgrade was approximately 15% that of a new simulator. This represents a significant cost saving.

In addition, the integrity of the MACS 6 facility and crew station has been preserved and is currently being used by five different programs for a variety of applications. This approach to meeting the needs of the JSF program has also facilitated additional JSF simulations being planned in MACS 6. Our low cost, reconfigurable solution has produced a better utilization of company assets, excellent reliability and a unique, reconfigurable dome visual system. Pilot and participant comments have all been positive. This facility has proven particularly ideal for carrier and vertical landing and takeoff evaluations. It is anticipated that this facility will continue to serve the needs of not only the X-32/JSF program but many others as well.

AIAA-99-4197

AUTOMATIC CORRELATED TERRAIN DATABASE GENERATION AND MANAGEMENT FOR GROUND VEHICLE SIMULATORS

Yiannis Papelis,
Shawn Allen,
Ben Wehrle

Simulator Technology Branch
National Advanced Driving Simulator & Simulation Center
The University of Iowa

Abstract

Modeling and interrogating the elevation of the terrain is necessary in both flight and driving simulation applications. In flight simulation, with the exception of ground based operations, the requirements for resolution and interrogation rate can often be met by the image generator or other simplified databases containing airports and the surrounding area. In ground vehicle simulators however, the requirements for interrogation rate, performance, resolution, and geographical coverage are much more stringent and separate databases are often used. Work described in this paper extends prior developments on correlated terrain databases specifically focused on driving simulation applications by adding key features on usability, correlated data generation, singularities, and performance. Key features of the earlier system included variable resolution storage, real-time execution time independent of the size of the database, and the ability to model vertically stacked terrain with minimal runtime overhead. Improvements of the new system include the automation of the database generation process given the source code of the visual databases, a new interrogation technique that eliminates distracting artifacts that appear across areas that have been sampled at different resolutions, and the use of graphical interactive tools for verifying the correlation of the final terrain database against the original visual model. The paper includes examples that demonstrate the process, along with information on the performance of the code under various conditions and computing systems.

Introduction and Prior Work

In ground vehicle simulators, where the interaction of the vehicle with the environment provides the majority of the feedback cues, a representation of the underlying terrain is critical for proper fidelity and realism.

Copyright © 1999 by Yiannis E. Papelis, S. Allen & Ben Wehrle. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

Terrain databases used for ground vehicle simulator applications have stringent requirements that eliminate the use of the Image Generator (IG) as a viable alternative for obtaining height above terrain (HAT) information. For example, a vehicle model executing at 240 Hz and requiring one interrogation at each wheel is equivalent to a 960 Hz interrogation rate for the terrain database. Given that at 240 Hz, the dynamic model has no more than 4.16 milli seconds of real-time available to perform all computations, the terrain interrogation must be very efficient, preferably taking no more than 1% of the total available time, or approximately 40 micro seconds. More complicated vehicle models often require even higher update rates [1] (i.e., 500 Hz or more) and multiple axis vehicles such as trucks require even more interrogations due to the larger number of contact points. Further increasing the requirements are sophisticated tire models that require the terrain elevation along a two dimensional patch as opposed to a single point [2], which in turn dramatically increases the number of interrogations that need to be performed per iteration of the dynamic model. In addition to the stringent performance requirements, ground vehicle simulator systems also necessitate the use of very high resolution data. Depending on the complexity of the tire model, resolutions as fine-grain as five or ten centimeters are necessary. Whereas visual databases often utilize textures to provide high resolution features, even though the underlying polygons are often larger than 125 meters, the actual terrain database must be able to represent high resolution features with distinct elevation and/or surface property feature sets. It is clear that the cumulative requirements of high fidelity ground vehicle simulation systems as described above necessitate terrain databases that are specifically designed to address these requirements.

Prior work on this area has focused on using regular grids for storing elevation and surface features, but with different capacities for dealing with large databases, and different interrogation algorithms [3, 4] for different applications. The approach described in [3]

revolved around the basic notion of using multi-resolution storage that allows optimal use of space for representing large databases with different resolution requirements. The basic concepts involved in the representation of the terrain are reviewed here but further details can be found in [3].

Terrain elevation, in its basic form, is provided by a uniform grid of elevation posts. Each post, often called a *dataset*, contains the elevation of the terrain at a given point along with information about the surface. The actual information varies with the application but typical items in simple applications include friction coefficients and various constants used to compute the forces upon the tire. In more complex applications, items stored in datasets include information about water bodies (i.e., depth, flow) along with soil properties at various depths. From the standpoint of the terrain database, each dataset represents a unit that is managed independent of its size or contents. Datasets are placed along a uniform grid within a rectangular area. The spacing of the grid can be different along the x-axis and the y-axis, but is uniform so that locating the four datasets that surround a query point is an operation that can be performed very efficiently and is independent of the size of the grid. The distance between datasets is the resolution. Note that in this context, the term *high*

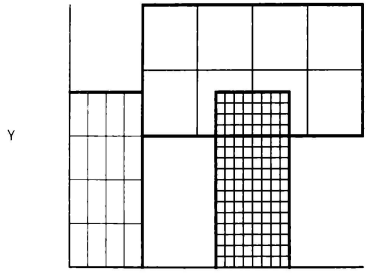


Figure 1 – Illustration of Datazones.

resolution refers to a smaller grid spacing providing more datasets per unit area. A group of datasets within a rectangular grid is called a *datazone*. Variable resolution storage is achieved by utilizing multiple datazones, each with a different resolution. Figure 1 illustrates a simple example of three datazones occupying different areas on the x-y plane. Note that datazones are rectangles whose sides are aligned with the x and y axis. Each intersection of lines within a datazone represents a location on the x-y axis that would be occupied by a dataset. So a datazones with n rows and m columns contains m * n datasets, and the dataset's x and y coordinates are implicit given its row and column position within a datazone.

Datazones can overlap. Overlapping datazones can represent overlapping terrain such as bridges or tunnels. When datazones overlap, it is not necessary for all of their datasets to provide terrain that is vertically spaced. In fact, it is possible to have overlapping datazones that contain data representing the same elevations. This often occurs when a datazone is larger than the elevated feature it represents, for example a bridge. The datasets representing the elevated feature will contain the new elevation, but the datasets that fall to the side of the bridge should in fact contain the same elevations as the datasets of the datazone representing the underpass.

Interrogation Algorithm

The input to the interrogation algorithm is a location on the x-y plane represented by a point P(x,y) along with an estimate of the z elevation. Thus the input to the query is a triple (x, y, z). There are three steps involved in determining the elevation of the terrain as follows:

1) Find the set of datazones that overlap point P.
2) For each datazone, do the following computations, storing the results internally:
   2.1) Compute the coordinates of point P with respect to the datazone origin, store in P'.
   2.2) Determine the row and column of the four datasets that surround P'.
   2.3) Use bilinear interpolation to compute the elevation at P'.
   2.4) Based on proximity to the four surrounding elevation points, pick the best dataset for representing the surface properties.
3) Given the initial estimate of z, pick the best candidate among the ones found during step (2).
4) Use bilinear interpolation to compute the normal vector at point P', given the best candidate.

The complexity of step (1) depends on the data structures used to aid the searching of datazones given the query point. In the worse case scenario, a linear search with O(N) complexity can be used (where N is the number of datazones in the system). Two dimensional search data structures can easily be used to reduce that complexity. Quad trees can reduce the complexity to O(log(N)), or as shown in [3], two dimensional hash tables can reduced the complexity to the point where it is independent of the number of datazones in the system, or O(1). The computational

complexity of Step (2) is independent of the total size of the database, the resolution, or the dataset size. For example, step 2.1 simply involves a few arithmetic operations (primarily subtractions), step 2.2 involves a few divisions and truncation operations, step 2.3 involves the bilinear formulas and finally step 2.4 involves few comparisons and data movements. The only O(N) complexity is due to multiple overlapping datazones. Generally, even though O(N) is not considered acceptable for real-time applications, the upper bound on N is the maximum number of overlapping surface in a datazone. In general, this upper bound is very low, usually no more than two or three, in off-road applications. Finally, steps (3) and (4) are also O(1) complexity involving few arithmetic operations that are independent of the size of the database.

In summary, provided that an efficient two dimensional search data structure is used for step (1) and a reasonably small bound can be placed on the maximum number of overlapping datazones, the interrogation algorithm has fixed order computational complexity thus is independent of the resolution of the datazones or the overall size of the database.

Runtime Management of Data

Given arbitrarily high resolution, terrain databases can often become too large to fit in the main memory of the computer running the dynamic model. In such cases, a runtime component is used to page in data necessary for terrain interrogation before they are needed by the dynamic models. The runtime component is using a prediction algorithm to estimate the datazones that will be needed by the dynamic model and load them in memory before the interrogation code actually needs them. A detailed analysis of the bandwidth requirement for various terrain types and densities was done at [3], but for clarity a small summary is repeated here. The data throughput required between the disk I/O subsystem and main memory to transfer data fast enough to keep up with a sweeping line whose length is equal to W, and is traveling at a velocity of v is equal to $n \cdot \frac{W \cdot v}{res^2} DSS$. The DSS symbol represents the size of the dataset, in bytes, res represents the resolution of the terrain, and n represents the average number of overlapping datazones over the sweep area. Note that the basic bandwidth increases in the presence of overlapping terrain. In applications where it is necessary to write back the data, such as when working with dynamic terrain, the bandwidth will double.

An additional consideration is the granularity of data as it is moved in and out of memory. If whole datazones are paged, there is significant potential for external fragmentation due to the varying size of datazones. If smaller transfer units, such as datasets are used, then the bandwidth will suffer and hard disks operate best when transfers take place in disk block units. Finally, using a transfer unit that does not correspond to a basic object in the terrain database, for example a 4096 byte block, then there is significant computation involved in determining the relation between the needed geographical coverage, and the actual data that has to be moved within the block.

Enhancements .

Despite its usefulness as an efficient terrain interrogation system, there are several limitations that became apparent following extensive use in various ground vehicle simulators. These limitations include the inability of the model to efficiently represent vertical edges, discontinuities across datazones of different resolution, and the challenging process by which such databases have to be constructed, especially when a corresponding visual database already exists. Work presented in this paper addresses these limitations.

Discontinuities due to elevation

One of the disadvantages of the elevation grid representation is that it cannot effectively represent vertical terrain profiles as often encountered in sidewalk edges or river beds. Generally, if the edge of such an object is aligned to the x or y axis, adjacent datazones whose datasets have different elevations can be used to represent the discontinuity.

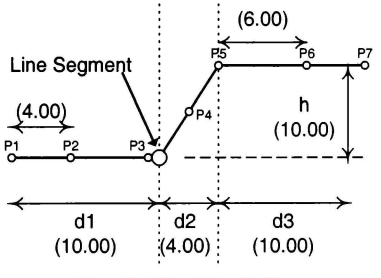


Figure 2 – Edge Object Profile.

When the features are not aligned to the x or y axis, vertical edges have to be represented by changes in the elevation on adjacent datasets. The maximum angle edge that can be represented by adjacent datasets is limited by the resolution of the datasets, with the extreme limit been a vertical edge which cannot be represented by elevation posts since it would require a

zero resolution. Unfortunately, such edges are very often encountered both in off-road and on-road applications. When off road, rock formations or river beds often require modeling vertical edges. Even though our focus is for off-road applications, it is interesting to note that even when on-road, curbs and sidewalks are very common and would pose the same limitations on any terrain model that depends on uniformly placed elevation posts.

To address this limitation an added layer of special effect *edge objects* are used to represent such terrain. An edge object consists of a series of connected line segments that all lie on the x-y plane, each segment representing the edge of a discontinuity of the terrain elevation. The discontinuity is represented by several parameters: d1, d2, d3, h and a base height. Figure 2 illustrates how these parameters model the profile of an edge. In that figure, a line segment would be perpendicular to the page. The base height establishes the elevation of the bottom area of the edge object. This allows the modeling of uneven terrain with arbitrary drops and independent of the resolution of the underlying elevation grid. Note that selecting a value of 0 for the parameter d2 yields a vertical edge.

The interrogation algorithm described earlier has to be slightly modified to accommodate edge objects. When the position being queried for is inside the d1 segment of the edge object (point P2), the terrain query interpolates between the height of the terrain grid and the base height of the object. At point P1, the query returns the height of the terrain grid, and at point P3, the query returns the base height of the edge object. When the position being queried for is inside the d2 segment of the edge object (point P4), the terrain query interpolates between the base height and the top height of the edge object. Finally, when the position being queried for inside the d3 segment of the edge object (point P6), the terrain query interpolates between the top height of the edge object and the height of the terrain grid. At point P5, the query returns the top height of the terrain grid, and at point P7, the query returns the height of the terrain grid.

When the line segment defining the edge object does not lie in the x-y plane the terrain query determines the elevation in two steps. It first assumes that the object does lie in the x-y plane and performs the same calculations as before. It then adds to this calculation a linear interpolation between the lower and upper points of the line segment to account for the segment.

Datazone Boundary Discontinuities

When traveling across datazones with different resolutions, the sampling of the terrain may yield different results on the edge, depending on which datazone was used. This problem is amplified in areas where the resolution is not high enough to capture all the effects of the underlying terrain. Whereas small errors due to under-sampling are acceptable, the discontinuities due to different datazone resolutions are not acceptable as they often cause jerks to the steering wheel or force the generation of sharp accelerations by the simulator's motion system. To resolve such errors the transition from the boundary of the current datazone to that of the next must be made more gradual. When a query is made near the edge of a datazone that is adjacent to a datazone with a higher resolution, a special smoothing interpolation is used as follows. Two new elevation posts are dynamically created at a distance consistent with the grid of the higher resolution datazone, but within the datazone with the lower resolution. The elevation is then determined using bilinear interpolation using the new "virtual" elevation posts of the current datazone. The actual interrogation results are computed using the two virtual datasets along with the datasets from the adjacent datazones.

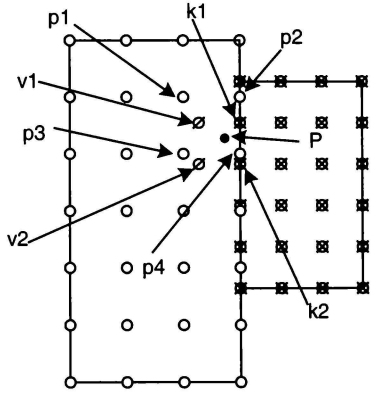


Figure 3 – Virtual Elevation Posts.

This process is illustrated Figure 3. Two adjacent datazones have different resolutions. The one on the right has a higher resolution than the one on the left. Let us refer to the resolution of the left datazone as Rl and the resolution of the right datazone as Rr. An interrogation is made at point P, which falls on the datazone on the left, within the quad defined by points p1, p2, p3, and p4 or any other adjacent dataset on the left datazone. That point is located less than Rr units away from the edge of the right datazone, in effect

triggering the generation of two virtual elevation posts. These new virtual posts, v1 and v2 are computed by interpolation between p1, p2, p3, and p4 but on a location that is consistent with the spacing of the high resolution datazone on the right. Once points v1 and v2 are calculated, the interrogation is performed using the quad consisting of points v1, v2, k1 and k2. Note that by the nature of the bilinear equation constraints, queries along a shared edge of two adjacent quads yield similar results no matter which quad was used. Use of the virtual elevation posts ensures that the transition between the two datazone becomes continuous even when there are large differences between the sampling rate of adjacent datazones. One disadvantage of this technique is that additional data structures have to be used to ensure that the computational complexity of the interrogation algorithm does not suffer due to the need to determine if adjacent datazones exist. Currently, a prototype system is under development and preliminary results indicate that use of simple adjacency lists provide a very efficient method for finding adjacent datazones. In this context, an adjacency list is a list associated with the edge of each datazone containing any adjacent datazones with higher resolution. The two additional bi-linear interpolation calculations are additive in nature, in effect increasing the computation time by a fixed amount, but they are still independent of the total size of the database.

Automatic Generation of Terrain Databases

One of the key challenges in the effective utilization of a terrain database system is the ease by which a terrain database can be constructed, and the final degree of correlation between the terrain data and the original visual database used as a source. The concept behind the automated terrain database extraction is simply to interrogate the visual database polygons and extract the elevation grid which is used at runtime. The actual process involves quite a few translation details along with various algorithms for determining the datazone layout and elevation post contents. These issues are further described below.

A block diagram of the terrain database generation process is shown in Figure 4. The process begins by extracting a list of polygons from a visual database. Only polygons representing the terrain skin are considered at this point. The actual method by which terrain polygons are discriminated from other polygons that represent features varies depending on the format of the source file and various implementation issues, but most formats provide for simple methods by which the actual polygons and their attribute information (i.e., texture names, etc.) can be dumped into an ASCII file.

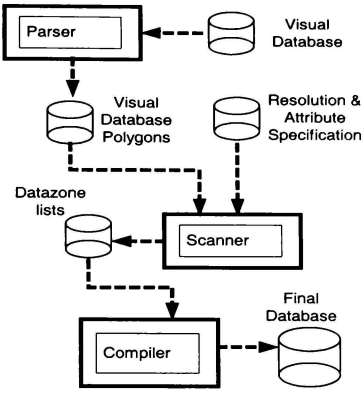


Figure 4 – Terrain Database Generation Process.

The current prototype incorporates parser implementations for the Evans & Sutherland General Database Format (GDF) and the MultiGen OpenFlight format. Additional formats, such as OpenInventor, or Sedris can easily be accommodated if necessary. The visual database polygons are triangulated (if necessary) and stored in a *TRI* file. Each line in this file contains the x, y, and z coordinates of each of its vertices, and any type specific information. Type information includes the surface properties associated with a triangle, or additional meta-data that may affect its interpretation. For example, lake beds or polygons representing water surface are identified as such allowing proper interpretation and storage in the dataset.

The user must also supply a datazone layout, or *DZL* file. A DZL file is a list of boundary-aligned rectangles along with a required resolution that describes the proposed datazones for the final datazone. Each line of the file contains the x and y coordinate for the lower left and upper right corners of the datazone, and the number of divisions along both the x and y axis. At the prototype system, this file is generated by hand, but work is underway for automating this phase of development with automated tools that use heuristics and optimization algorithms for automatically determining an optimal, based on user specified criteria, datazone layout.

The tagged triangles and DZL file are fed to the scanner program which is responsible for generating the datazones and associated datasets. The scanner operates

as follows. The polygons are first loaded into memory from the TRI file, and their projections onto the x-y plane are stored in a quadtree lookup table. The terrain is then broken up into datazones as specified by the DZL file, and each datazone is broken up into a grid of datasets, or elevation posts. In order to construct the final datazone the elevation, and the type specific information must be determined for each elevation post within the datazone. For each elevation post the quadtree returns all polygons and objects that contain the coordinates of the elevation post within its boundaries.

One of the challenges facing the scanner is the proper generation of overlapping datazones when the multiple polygons, each representing a different overlapping surface, exist within the boundary of a datazone. To detect and label overlapping triangles the sampler keeps a reference of the triangle used to determine the elevation of the previous elevation post. The sampler also maintains a list of any triangles that have been tagged as overlapping within the datazone. The algorithm of selecting which triangle to use to determine the elevation of the post proceeds as follows:

1) The quadtree returns all triangles that contain the elevation post

2) Any triangles in the quadtree list that are also in the overlapping list are removed from the quadtree list.

3) If any of the remaining triangles match the triangle from the previous elevation post

A) Select that triangle
else
B) Select the triangle with the lowest elevation

4) Use the selected triangle to determine the elevation and characteristics of the post

5) Remove the selected triangle from the quadtree list

6) Any triangles that remain on the quadtree list represent overlapping terrain and are added to the overlapping list. As an exception to this step, if the elevation of a triangle in the modified quadtree list is less than some specified tolerance from the elevation of the selected triangle, then they are said to share the elevation post and are not considered overlapping. This allows multiple triangles to share a common vertex or line segment.

If the overlapping list contains at least one entry once the characteristics of all of the elevation posts have been determined then the datazone contains overlapping terrain. At this point the sampler creates a second quadtree that contains only those triangles that were on the overlapping terrain list. Once the new quadtree has been created the overlapping terrain list is cleared so new overlapping triangles may be detected. A new datazone is also created with the same bounding box and elevation post densities as the previous one. The new datazone then queries the new quadtree to determine the characteristics of its elevation posts. The same steps as before are used to select triangles, and the process continues until the overlapping terrain list is empty at the end of a datazone. At this point all levels of overlapping terrain have been modeled and the sampler continues to the next datazone in the DZL file.

Figure 5 illustrates an example. In this case, neither triangle A nor triangle B are on the overlapping terrain list, but triangle A matches the previous triangle so it is used to determine the elevation of the post.

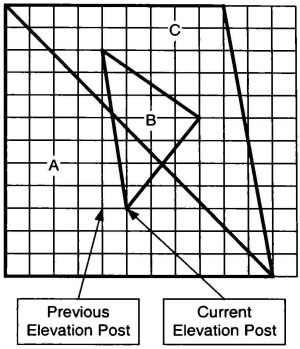


Figure 5 – Example 1.

Since triangle B is not overlapping on a vertex or line segment, it is placed on the overlapping terrain list for the next post.

Figure 6 illustrates the operation of the algorithm during the generation of another pair of elevation posts. In this case, triangle B is on the overlapping terrain list and is removed from the quadtree list. Triangle A matches the previous triangle so it is used to determine the elevation of the post.

The final step in the process of generating the terrain database is the compilation step. The compilation step is responsible for pre-computing all the internal data structures used during the interrogation algorithm for searching applicable datazones. Also, the adjacency lists which are used to determine if virtual elevation posts need to be computed are also created during this step. Finally, datazones that exceed the maximum size required by the lookahead algorithm are broken into multiple smaller datazones in such a way as to minimize internal fragmentation. The input to the compiler is simply a list of datazones and associated

datasets, and the output from the compiler is an integrated terrain database file that can be used as input to the interrogation code, or can be automatically managed by the run time lookahead loader for inclusion in a real-time simulation system.

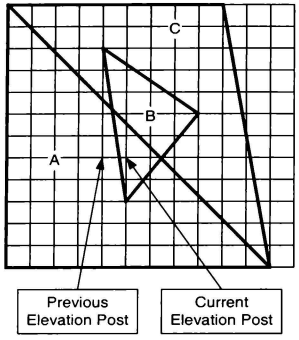


Figure 6 – Example 2.

Implementation Issues and Status

The system described in this paper is currently under construction. Several prototype implementations have been coded to experiment with execution speed using various methods for the two dimensional lookup functions, and for validating the expected size independent execution speed. At this point, the final system is nearing completion, with several of the components already coded and tested. A parser for the E&S GDF format has been coded and tested and an OpenFlight parser is underway.

One of the challenges encountered during the implementation of the GDF parser is parsing databases that are programmatically specified. Specifically, databases that use *basis sets* provide a list of templates that during compilation of the visual database are placed at different places in the database while at the same time adjusting their vertices to snap onto existing terrain. This feature minimizes modeling time since it allows the re-use of cultural features and vegetation templates across large areas without the need to manually apply each and every feature. However, use of basis sets eliminates the ability of a tool to extract the terrain skin from the source code of a database. Instead, the parser tool will have to operate on the final binary file as prepared for the image generator, which already contains the instances of the various templates as they will appear on the IG. Some additional corrections have to be made to adjust the triangles to accommodate their modified position due to the snapping onto the surrounding terrain, but with these modifications, the tool performs as expected.

The scanner program is also completed and currently undergoing testing. To evaluate the operation of the algorithm that discriminates overlapping terrain and produces multiple datazones to represent vertically stacked terrain, several reference databases were created. One of the most challenging ones is the representation of a multi-level parking ramp as shown in Figure 6. In addition to providing a challenging test for the scanner, this reference database allows us to determine the sensitivity of the execution time to multiple overlapping surfaces.

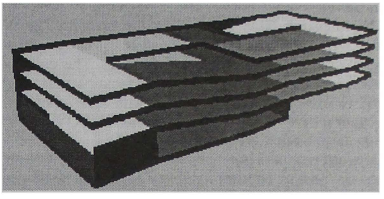


Figure 6 – Reference Database for Scanner Tests.

Finally, the compiler and interrogation code have been coded and tested. The runtime lookahead manager is currently under development.

Performance

The software for the off-line tools is written in C++ and has been compiled and tested in several Unix and PC platforms including IRIX, SunOS, Linux and WinNT. Similarly, the interrogation code is written in C and is highly portable. To this date, it has been compiled in all the systems that the off-line tools are operational, but in addition, it has been compiled in the VxWorks operating system as part of simulators whose software runs on embedded system using that operating system.

The most critical performance number is the execution time of the interrogation code, which along with the lookahead loader is the only component that runs in real time. The assumption that the execution code is independent of the size of the database has been experimentally validated. A set of experiments were run to measure the execution speed of the interrogation code. The total database size is approximately 50 mega-bytes, and the datazone search algorithm (described in step (1) of the algorithm) is using a data structure that searches across 10 datazones before finding the applicable ones This is a very conservative estimate since in practice, the search is often limited to no more than three or four datazones. Several configurations were tested. In the first configuration,

all query points were on areas of the database that had no overlapping terrain. In the second configuration, the interrogation point was on a location containing two overlapping surfaces. The difference in execution time between these two cases represents the computational load of the bilinear interpolation calculations. Another factor entering the experiment is compiler optimizations. Generally, depending on heavy compiler optimizations makes the software less portable, however, due to the nature of the bilinear interpolation, the potential for severe pre-calculations, and the ability of better using the pipelines often encountered in modern processors, the experiments did include measurements of compiler optimized code.

The following table summarizes the execution times that were obtained per interrogation. Multiple tests were run, and the average is reported. The column labeled NOOV, NOOPT contains execution times for no overlapping surfaces and with no compiler optimizations. The column labeled NOOV, OPT contains execution times for no overlapping surfaces but with compiler optimizations turned on. The column labeled OVLP, NOOPT contains execution times for two overlapping surfaces with compiler optimizations turned off, and finally, the column labeled OVLP OPT contains execution times for two overlapping surfaces with compiler optimizations turned on. No special effect objects were used in this experiment. All units are micro seconds (1E-6 seconds).

	NOOV NOOPT	NOOV OPT	OVLP NOOPT	OVLP OPT
Mips R44k, 250M	12	6	19	9
Mips R5k, 195M	11	5	17	8
Mips R10k, 195M	8	4	11	6

Table 1 – Interrogation Performance.

Conclusion and Future Directions.

This paper describes an integrated process for generating, managing and interrogating variable resolution terrain databases used primarily in ground vehicle simulation applications. Various algorithms and techniques for generating data that can be for the real-time interrogation of the terrain at very high speeds were described along with the rational for the various chosen design decisions.

Several areas can benefit from improvements in the existing processes. Generating the DZL files is at this point a manual processes, sharply contrasting the automation found in the remaining of the system. Furthermore, even when given enough time, it is hard to manually estimate an optimal datazone layout. Clearly, a much needed improvement is the development of a tool that based on some optimization criteria can automatically generate an optimal datazone layout. Some of the criteria may include minimizing the overall storage requirements, minimizing the sampling error, or minimizing the number of datazones.

Another area that can benefit from additional work is the determination of the surface properties, once a visual database polygon has been identified as the underlying terrain skin. Currently, the surface properties are calculated based on a lookup table that given the texture and/or color of the triangle, provides the necessary surface properties. However, a popular visual database modeling technique is using global textures for adding details without using too many polygons. Global texture is a term that refers to texture that contains features drawn within it. For example, a single texture may contain a lake, roads, and forests, however, given the current technique, all queries within a polygon using that texture would be associated with the same surface properties. An alternative would be to determine the actual texel contained on the location of the elevation post and the use the texel value to determine the surface properties associated with that elevation post. Finding efficient ways and developing manageable logistics to map texel values to surface properties is an open issue at this point.

References

[1] D. Solis, E. Haug, and S. Shome, "Advanced Numerical Techniques for Hi-Fidelity On- and Off-Road Vehicle Simulation", to be presented in the MST Conference, Portland, Oregon, August 9-11, 1999.

[2] D. Solis, E. Haug, and LD. Chen, "Virtual Prototyping of Wheeled and Tracked Vehicles", to be presented at the All Electric Combat Vehicle (AECV) Conference, Bonn, Germany, September 21-23, 1999.

[3] Y. E. Papelis, "*Terrain Modeling Issues for High Fidelity Ground Vehicle Simulators*," 5th Annual Conference on AI, Simulation and Planning in High Autonomy Systems, Dec 7-9, 1994, Gainesville, FL, pp. 48-54, 1994

[4] T. Stanzione, "Suitability of the Standard Simulator Interchange Format for Representation of Terrain for Computer Generated Forces," In Proceedings of the Fourth Conference on CGF and Behavioral Representation, pp. 231-238, May, 1994.

[5] J. G. Kuhl, D. Evans, Y. Papelis, R. Romano, G. Watson, "*The Iowa Driving Simulator: An Immersive Environment for Driving-Related Research and Development*,", IEEE Computer, 28(7) pp. 35-41.

THE KARI ENGINEERING SIMULATOR : A DESIGN TOOL FOR REGIONAL TRANSPORT AIRCRAFT DEVELOPMENT

Kapseong Ro*, Sanghyuk Lee† , Inkyu Kim‡ , Vladymyr Povstyanoy§ & Chul-Ho Lim¶

Korea Aerospace Research Institute (KARI)
P.O. Box 113, Yusong, Taejon, 305-600, Korea

ABSTRACT

This paper describes the engineering simulator developed by Korea Aerospace Research Institute located at Taejon, Korea. The main purpose of this simulator is to carry out research and development work in developing a medium range transport aircraft. The simulator is composed of low cost, high fidelity subsystems and components capable of complying the top-level requirements by the regulatory body. These components are interconnected via high-speed data bus and network, while operating with the in-house developed real-time flight simulation software. Throughout the development process, much of the design emphasis is concentrated upon flexibility, expandability and modularity of the system software and hardware. In this paper the overall aspects of the KARI Engineering Simulator development program are presented, that include the discussions on the use of flight simulator for aircraft design, the design requirements and objectives, and the features of simulator subsystem. Also discussed are the further development in progress, some of the ongoing research work and the future plans.

I. INTRODUCTION

The design and development of a new airplane and its systems are very complex and iterative process including theoretical, computational and experimental methods. The use of piloted simulator is recognized as an essential design tool for the most part of aircraft design stages because it allows to solve complex, interdisciplinary problems that may not be easily solved by any other methods. Some of these problems were studied, as presented in the next section, prior to the development of the simulator. These studies led to formulate the practical design requirements and objectives (DR&O), which required to build a full flight simulator capable of simulating cockpit instruments, visual, sound, motion and control loading characteristics of aircraft in both ground and in-flight operation. Currently the simulator is developed, except motion simulation capability as shown in Figure 1, and in use for carrying out research to study a candidate control system model. This paper presents the general aspects of the development process, system features, and the software design for the Engineering Operator Station.

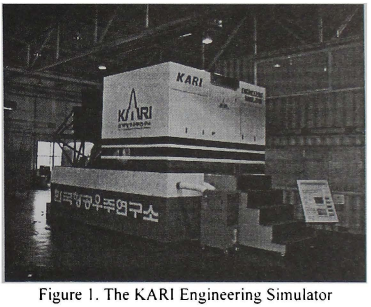


Figure 1. The KARI Engineering Simulator

II. FLIGHT SIMULATOR IN AIRCRAFT DEVELOPMENT

A piloted-simulator is an integrated design tool for the design and development of a new aircraft, since it not only provides the analytical results of aircraft performance, stability and controls characteristics, but also provides 'man-machine' characteristics of a vehicle in the design process. Some of important tasks in the aircraft design that can be effectively conducted using a piloted simulator are as follows:

---

*. Senior Researcher, Aircraft Division
†. Researcher, Aircraft Division
‡. Researcher, Aircraft Division
§. Visiting Scholar, Aircraft Division
¶. Director of Aircraft Division

Copyright © 1999 by the American Institute of Aeronautics and Astronautics Inc. All rights reserved.

1. Investigation of the influence due to the aircraft parameter variations:
   - minimal degree of longitudinal/lateral static stability for selecting minimal dimensions of horizontal and vertical tail.
   - influence of the wing parameters on lateral flight characteristics (Dutch Roll – Spiral stability) for selecting their optimal relationship.
2. Selection of the airplane primary control system structure and parameters:
   - selection of control lever types, i.e. conventional column/yoke/pedal, mini-stick or side-stick.
   - dimensions, arrangement, movement and forces applied to the controls.
   - pilot opinions on the controls feel.
3. Selection of kinematic characteristics:
   - design of the gear ratio of the control lever to the control surfaces
   - design of control methodologies based on the speed, flap position, horizontal tail, etc.
4. Determination of the necessity for implementing the stability and control augmentation system (SCAS).
   - system of stick force gradient variation, alpha & beta limiting system, normal & side load factor limit, flight speed limit.
   - selection of parameters and algorithms for operating these system.
5. Selection of parameters of trimming and balancing system; the response of control system in normal and failure situation.
6. Determination of operational mode of control system for failure situations or allowable configuration; indication of the control system serviceability and failure.
7. Selection of flap and slat operation algorithm.
   - rate of retraction/extension, flap retraction in go-around or according to flight speed.
   - selection of flap and slat retraction/extension at take-off and landing when the aircraft system such as engine, controls and hydraulics are at normal and failure situation.
8. Preliminary estimation of airplane and its systems corresponding to FAR-25 requirements.
9. Etc.

In general, the flight simulator is an essential design tool to carry out the research of:

- the aircraft flight dynamic characteristics
- the man-in-the-loop simulation (MILS)
- the hardware-in-the-loop simulation (HILS)

III. DESIGN PROCEDURES & SIMULATOR DESCRIPTIONS

There are many kinds of flight simulators depending upon their applications, and even within a single objective such as pilot training the range of simulators is quite broad. However, it is true that a human pilot is a key element for all cases, and the simulation fidelity is the main factor to obtain valid results. The fidelity of simulator is mainly dependent on the pilot perception. Generally speaking, up to 80% of the overall fidelity of a flight simulator may be achieved with a simulated cockpit instrument, out-the-window (OTW) visual, sound effect and motion system. Therefore, it was not an exception for the KARI Engineering Simulator to have visual, sound and motion system while a particular emphasis is paid upon the ability to reconfigure each system as well as the system as a whole. This was the basic design requirement in developing the KARI Engineering Simulator. The design family tree was then established as shown in Figure 2. Based on this family tree, the work-breakdown-structure (WBS) and its dictionary was made.[1] A simulated cockpit including primary and secondary controls were designed based on the result from the conceptual design of Korea Regional Transport and a typical 70-100 seat transport. Low cost commercial off-the-shelf (COTS) system including electronic flight display system, the OTW visual display system, sound system, control-loading system were procured and implemented. A visual image generator was developed using the Power Onyx2 with InfinityReality® graphics engine from Silicon Graphics Inc., and two major airfields in the country were also developed. The VME BUS and Ethernet are two means of interconnecting these subsystems, as shown in Figure 3. The features of these systems are briefly described.

Simulated Cockpit System

The simulated cockpit system is composed of three structural components of baseframe, cockpit shell and operator cabin. The baseframe was designed to mount a wide angle collimator (WAC) monitor type display system, and to incorporate conventional primary controls of column/yoke/pedal to be linked with force actuators. The layout of flight instrument was referenced to a typical 70-100 set transport as well as some of the results from the conceptual design study. The flight instrument is mainly a glass cockpit type made up of six set of 8 by 8 inch high resolution (1024 by 1024) cathode ray tube (CRT) monitors in addition to some of essential standby indicators. The glass cockpit system a stand alone system driven by a Pentium PC equipped with specially designed, PowerPC 604 based graphics board for each monitor. The six CRTs are used for a pilot and copilot electronic flight instrument system (EFIS), engine indicating and crew alerting system (EICAS) and

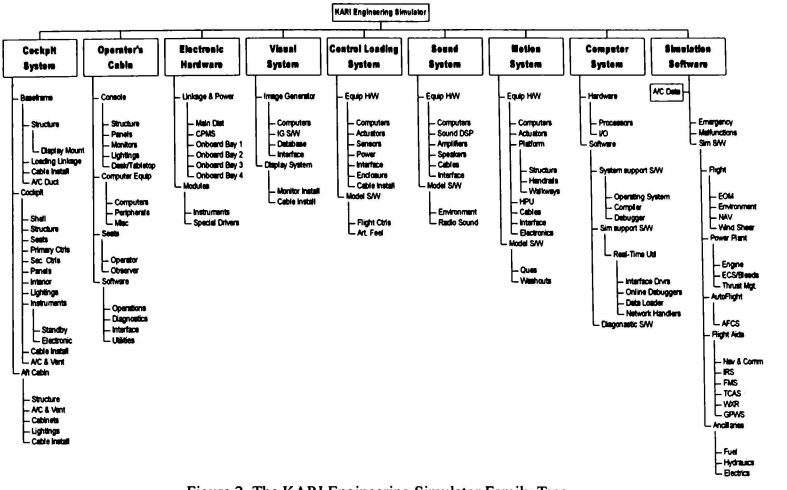


Figure 2. The KARI Engineering Simulator Family Tree

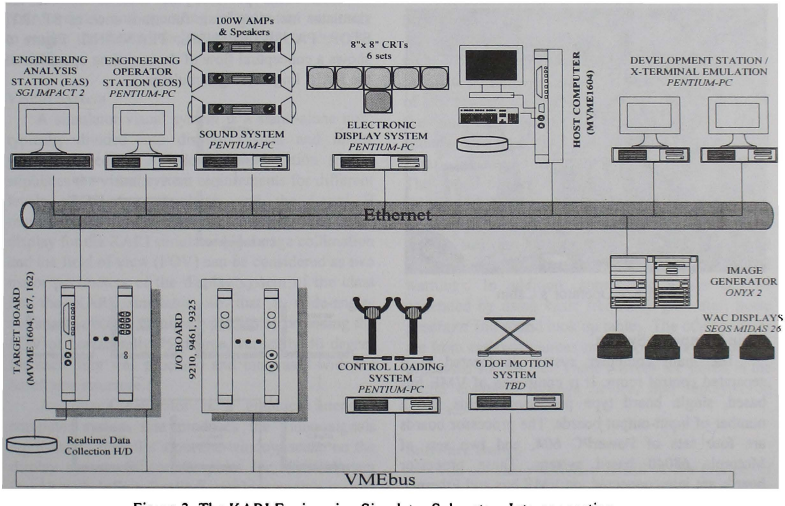


Figure 3. The KARI Engineering Simulator Subsystem Interconnection

electronic centralized aircraft monitoring (ECAM) system. The display contents of each system can be easily and quickly created and modified by using a menu driven, avionics design oriented software. The data link for this glass cockpit system to the host computer is Ethernet/UDP, and it graphically displays

current flight information in real-time. In addition to primary controls, some of secondary controls are manufactured including thrust lever, flap lever, trim knobs, land gear lever, fuel lever, speed brake lever etc. Figure 4 shows the figure of cockpit layouts. The operator cabin as shown in Figure 5 provides the space for operator station console, observation seats and onboard electronic cabinets used for the signal junction patch, power supply and distribution, and some of onboard equipment such as sound amplifiers, power amplifiers. etc.

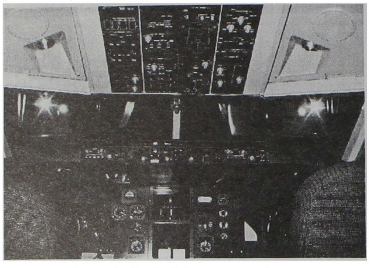


Figure 4. Cockpit Instrument layout

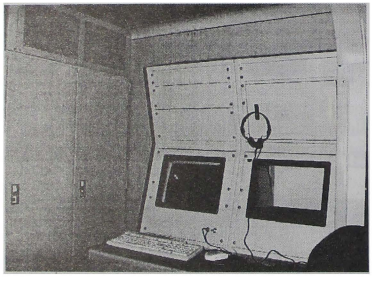


Figure 5. Operator's Cabin

Main Computer System

The main computer system is located in a separated control room. It is composed of VME bus based, single board type processor boards and a number of input-output boards. The processor boards are four sets of PowerPC 604, and two sets of Motorola 68040 based system. These processor boards are interconnected via VME bus and Ethernet, and operates in 'host-target' configuration. One of Power PC 604 based processor board is designated as the host computer that is equipped with a general purpose operating system (UNIX/X Windows), a data storage device and a real-time kernel. The rest of processor boards are assigned to perform flight dynamics computation, input/output operation, network operation and real-time data collection on a data storage device. There are several VME bus type ADC (analog to digital converter) DAC (digital to analog converter) and DIO (digital input-output) device providing approximately 800 channels to handle various cockpit input and output signals. All operations in the host computer are executed in a timely, prescribed manner under the control of simulation manager program. The simulation manager program is basically a traffic controller developed using the real-time kernel routines.[2] It monitors and controls the simulator among several tasks which share the common information. For subsystems that requires Ethernet communication link, the simulation manager acts as a server program to transfer the data on demand. Flight dynamic computation is the top priority task that updates the common data structure called 'shared memory' at the rate of 10 m/s. A Pentium PC is designated as the Engineering Operator Station (EOS) computer to control the simulator operations. A graphical user interface (GUI) program, called the operator, is a client program to the simulation manager via Ethernet TCP/IP, which provides the user to control the simulator including basic functions such as START, STOP, PAUSE, RESUME, PLANNING. Figure 6 shows a conceptual flow of the real-time operation of the KARI Engineering Simulator.

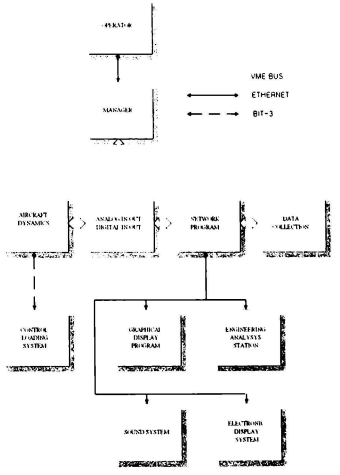


Figure 6. Real-time Software Structure

Control Loading System

The primary controls in the simulated cockpit is linked with electric torque motor based force actuator system. The system is manufactured by Science Application International Corporation (SAIC) in the U.S. It is composed of a Pentium PC with proprietary input/output boards, electric torque motor and force sensor for each channel of roll, pitch and yaw axis. The system comes with generic control system models of which many of model parameters can be modified to realize different feel characteristics. A high-speed bus-to-bus communication device is used to transfer the data between the main computer system and the control loading system computer.

Design and assembly of mechanical linkage between the primary controls and control loading actuators was a very challenging problem. The control system model in the control-loading computer is realized at the end of actuator, and the pilot would not generally feel these characteristics on his controls due to linkage mechanism between the actuators and the controls. This is because the linkage mechanism itself adds some dynamic effect in addition to the realized model. In general, the construction of linkage mechanism must follow the rules and specifications for the actual airplane case, and careful calibrations of both hardware and operating software are necessary to realize the desired feel characteristics at the pilot controls.

Visual System

A simulator visual system is a stand-alone unit, typically divided into display system and image generator. Federal Aviation Administration (FAA) stipulates the visual system requirements for different levels of training simulator, and the Level B requirements were applied in selecting the visual display for the KARI simulator [3]. Image collimation and the field of view (FOV) can be considered as two major parameters of the display system of the class for the KARI simulator, so that a wide-angle-collimated (WAC) monitor type display providing the FOV of horizontally 75 degree, vertically 40 degree for each pilot was procured and integrated with the baseframe structure.

An image generator is a graphics intensive computer system that produces the video signals representing the pilot's out-the-window scene on the display system. Silicon Graphics Inc. (SGI) Power Onyx2 with InfinityReality® graphics engine is used to design the image generator of the KARI simulator. It is still in the process of being developed, but it is adequate in terms of carrying out immediate research and development work. Currently, it provides required FOV of horizontally 75 degree, vertically 40

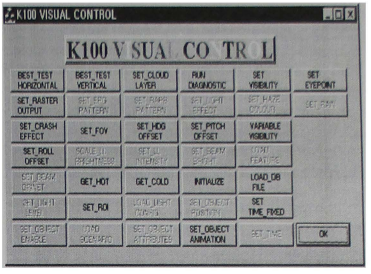


Figure 7. Visual Command Window on EOS

degree in 800 by 600 resolutions, three channels mode with the update rate of 30 Hz. It communicates with the host computer via Ethernet/UDP. The user selection functions such as visibility adjustment, time of day and runway selection, height of terrain return, cloud bottom/ceiling selection are available in the Engineering Operator Station (EOS) computer as shown in Figure 7.

Sound System

The sound system used in the KARI Engineering Simulator is a commercial-off-the-shelf product from Advanced Simulation Technology Inc. in the U.S. An eight channels digital signal processing (DSP) card is installed in Intel 486 based personal computer. Total of ten speakers are implemented inside and outside of the simulated cockpit to generate various flight sounds, navigation and communication sounds and aural warnings throughout the entire flight envelope. The sound system computer is communicating via Ethernet/UDP to the host computer. Sound cues can be created from two sources, one from the model builder software for engines of different types and numbers, navigation and communication and aural warnings. In addition, appropriate noises can be generated by using basic frequency generator, noise generator and sound look up tables. The other sources are from external sources such as prerecorded sounds, processed sounds, etc. These sources are stored in the sound computer storage device, and replayed on demand.

Software Design

The simulator software is still in the process of developing and being improved. The software design for the KARI Engineering Simulator can be generally divided into two groups – system operation software and aircraft simulation software. The operation software includes real-time operation programs such as simulation manager, input/output operations,

subsystem interconnection/operation software, the Engineering Operator Station computer operation, etc. The aircraft simulation software includes all programs related to aircraft and its system dynamic models, procedures, etc. Throughout the development process, much effort was being paid upon for flexibility and modularity of the each program.

The aircraft simulation software design requires comprehensive data for an aircraft to be simulated. These data are categorized as follows:

- configuration/design data
- simulation modeling data
- checkout data
- flight test validation and proof of match data
- system verification data.

At present, the aircraft is still in its early design stage and the simulator is an engineering development purpose, configuration/design data and simulation modeling data are essential. These data are frequently changed and comparisons for these changes are important, so that the simulation data for each aircraft subsystem must be in the form of database and able to be selectively loaded by an operator for each simulation session.

IV. DEVELOPMENT IN PROGRESS

Much of development work so far had been concentrated upon the construction of hardware and integrating them with the software. Recent effort in progress is paid upon the refinement and restructuring of the simulator software to improve the user interface and to increase flexibility for engineering purpose.

Both for training and research simulator the design of operator station has been an important design issue [4], and the Engineering Operator Station (EOS) is naturally a major area for improvement. Unlike training simulators, the KARI Engineering Simulator is required to accept frequent changes in model structure and system data, to provide efficient ways to handle simulation models, to manage simulation output results for pre and post analysis, etc. These functional capabilities should be systematically reflected into the Engineering Operator Station software through graphical user interface.

In order to design an EOS user interface and to restructure the aircraft simulation software, an abstraction of simulation as a design methodology is made in terms of three iterative processes – modeling, simulation, analyses of simulated data as depicted in Figure 8. Each of these stages has an intermediate, but associated targets. A target for modeling is to develop a model based on available modeling data. A target for simulation is to produce simulated output data from the model. A target for analysis is to

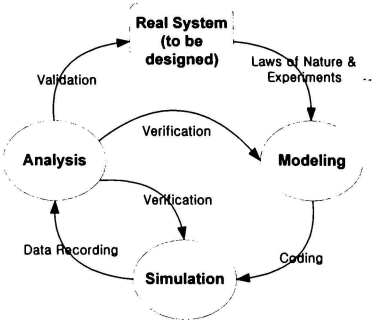


Figure 8. Simulation as a design methodology

provide a final conclusion based on the simulated data to be transferred to the design of the real system. This abstraction is used as a skeleton to design the EOS software. Figure 9 shows the main user interface window on the EOS of the KARI Engineering Simulator.

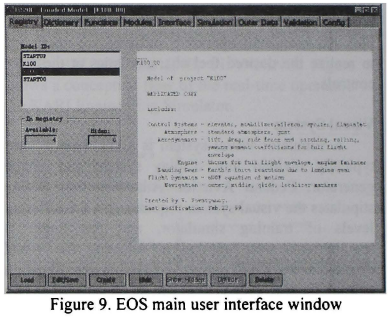


Figure 9. EOS main user interface window

The main window starts with 'Registry' page where information for different models are stored. All components of a model are placed in a directory named by the same as the model name. During a simulation session, any model in the registry can be loaded as an active model by 'Load' button in the footnote area. In addition to default 'Registry' page, it has pages of 'Dictionary', 'Functions', 'Modules', 'Interface', which provide the modeling capabilities while 'Simulation', 'Outer Data', 'Validation' and 'Configuration' pages provide the means to carry out simulation and data analysis. In Figure 10, 'Dictionary' page is illustrated, where a designer can edit, save and delete simulation variables and parameters using the corresponding buttons in the

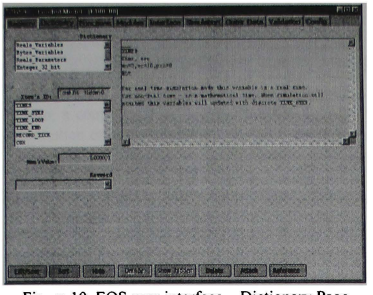


Figure 10. EOS user interface – Dictionary Page

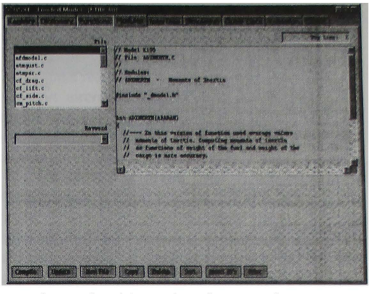


Figure 12. EOS user interface - Modules Page

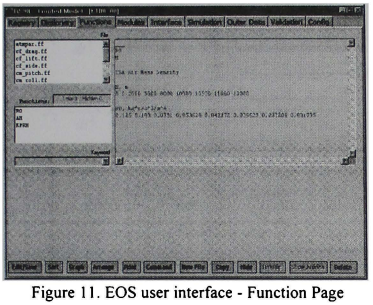


Figure 11. EOS user interface - Function Page

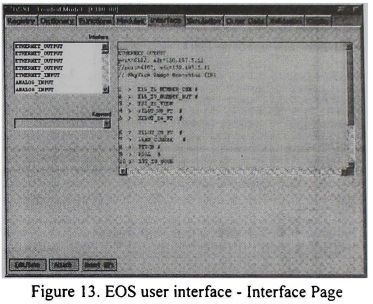


Figure 13. EOS user interface - Interface Page

footnote area.

Figure 11 shows the 'Function' page where functionally dependent data files are created, stored and edited. These data are commonly represented as a tabular form representing aerodynamic data for airframe and control surfaces, engine data, landing gear data, etc. These data can be plotted and printed with corresponding buttons in the footnote area. The command button in this page executes special operations on these data files such as computing a specific function data for given values of arguments, creating new functionally dependent data files from already existing files, approximating for given range of arguments, etc.

In 'Modules' page as shown in Figure 12, program components of models can be created, edited and compiled. The user can easily edit a C-program representing the model associated with the functionally dependent data files created in the 'Functions' page. It is then complied in the same window by an external C-compiler. An example is shown in Figure 12 illustrating aerodynamic drag computation.

In 'Interface' page as shown in Figure 13, simulator hardware and subsystem interfaces including I/O system and Ethernet lines are centrally controlled and monitored.

After setting up the models with the pages mentioned above, the next group of pages provides the simulation and analysis capabilities. This starts with 'Simulation' page as shown in Figure 14. In this page an user can create and edit a simulation scenario of interest. After creating the simulation scenario, the user controls the simulation with tool buttons. The tool buttons are predefined functions such as initialization, start, stop, suspend, real-time /non-real-time mode, record on/off, plotting options window as shown in Figure 14. The plotter is the default for visualization in the simulation page, and it can be customized with the plotter options window as shown in Figure 15.

In 'Outer Data' page and 'Validation' page, the user may carry out necessary operations, such as managing and visualizing data, on the simulated data as well as the real flight tested data. Mathematical tools such as parameter identification for stability derivatives may be incorporated in this page.

In 'Config(uration)' page as shown in Figure 16,

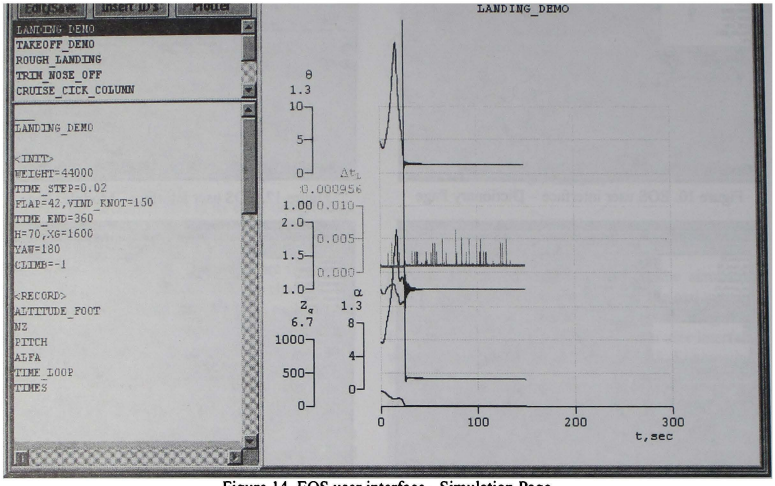


Figure 14. EOS user interface - Simulation Page

the user can configure a simulation with or without subsystems operation.

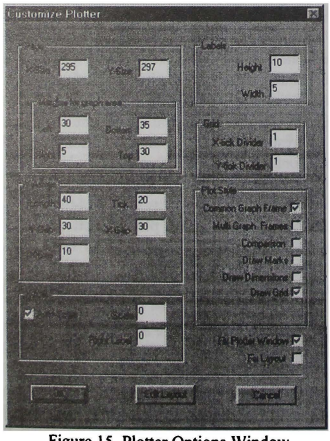


Figure 15. Plotter Options Window

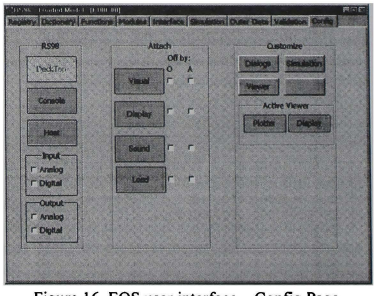


Figure 16. EOS user interface – Config Page

V. FUTURE RESEARCH & PLANS

One of the recently considered research topics on the application of the KARI Engineering Simulator is to study the aircraft flight characteristics in case of failure of its own or other system such as engines and flight controls mechanism. This is a very important problem of the safety of flight, and one of most

practical and important problems of the engineering simulator application for aircraft design.

An off-line simulation study is carried out for a situation of control surface linkage failure. Though many of modern transport adopt a fly-by-wire control system, a control system designer may obtain useful information on the aircraft response by considering a typical cable-linkage based control system as shown in. Figure 17. Figure 18 shows a hypothetical idealization in order to derive the equations of motion. The governing equations of motion for the elevator control system at normal operation, and for the failure case of right elevator control cable can be expressed as below:

Control System EOM at Normal Operation

$$J_1 \cdot \frac{d^2\varphi_1}{dt^2} = F_1 \cdot r_1 - K_\varphi \cdot (\varphi_1 - \varphi_2) - K_1 \cdot (r_2 \cdot \varphi_1 - r_3 \cdot \delta_{e_1}) - C_{f_1}(\dot{\varphi}_1) \cdot r_2$$

$$J_2 \cdot \frac{d^2\varphi_2}{dt^2} = F_2 \cdot r_1 + K_\varphi \cdot (\varphi_1 - \varphi_2) - K_2 \cdot (r_2 \cdot \varphi_2 - r_3 \cdot \delta_{e_2}) - C_{f_2}(\dot{\varphi}_2) \cdot r_2$$

$$J_{\delta_{e_1}} \cdot \frac{d^2\delta_{e_1}}{dt^2} = K_1 \cdot (r_2 \cdot \varphi_2 - r_3 \cdot \delta_{e_1}) + M_{h_1}(\delta_{e_1}, \dot{\delta}_{e_1}, \alpha_{ht}, \delta_{trim}, \ldots)$$

$$J_{\delta_{e_2}} \cdot \frac{d^2\delta_{e_2}}{dt^2} = K_2 \cdot (r_2 \cdot \varphi_2 - r_3 \cdot \delta_{e_2}) + M_{h_2}(\delta_{e_2}, \dot{\delta}_{e_2}, \alpha_{ht}, \delta_{trim}, \ldots)$$

Control System EOM at Failure Situation

$$J_1 \cdot \frac{d^2\varphi_1}{dt^2} = F_1 \cdot r_1 - K_\varphi \cdot (\varphi_1 - \varphi_2) - K_1 \cdot (r_2 \cdot \varphi_1 - r_3 \cdot \delta_{e_1}) - C_{f_1}(\dot{\varphi}_1) \cdot r_2$$

$$J_2 \cdot \frac{d^2\varphi_2}{dt^2} = F_2 \cdot r_1 + K_\varphi \cdot (\varphi_1 - \varphi_2) - C_{f_2}(\dot{\varphi}_2) \cdot r_2$$

$$J_{\delta_{e_1}} \cdot \frac{d^2\delta_{e_1}}{dt^2} = K_1 \cdot (r_2 \cdot \varphi_2 - r_3 \cdot \delta_{e_1}) + M_{h_1}(\delta_{e_1}, \dot{\delta}_{e_1}, \alpha_{ht}, \delta_{trim}, \ldots)$$

$$J_{\delta_{e_2}} \cdot \frac{d^2\delta_{e_2}}{dt^2} = M_{h_2}(\delta_{e_2}, \dot{\delta}_{e_2}, \alpha_{ht}, \delta_{trim}, \ldots)$$

The results of off-line simulation are illustrated in Figure 19. As expected, the same amount of column input produces significantly reduced control actions. This kind of study is almost impossible to carry out in real test flight, but the characteristics and the recommendation for emergency procedures must be prepared.

At this time, a couple of emergency flight situation is in the process of being implemented for the real-time man-in-the-loop simulation.

The developed KARI Engineering Simulator is the national resource for research and development of transport aircraft. The system is now adequate for performing essential studies in a prototype develop, but requires continuous improvements in several aspects.

Not so distant future, the KARI Engineering Simulator will be equipped with a motion system in order to enhance the simulation fidelity for the handling qualities evaluation of designed aircraft. In addition, a study is planned for hardware interface methods such ARINC 429 serial interface and MIL-STD-1553B communication protocols in order to expand the research capabilities for hardware-in-the-loop simulation test.

REFERENCES

[1] Ro, K., Park, J.W., Lee, S.H. and Kim, I.K., "Work Break Down (WBS) Structure and WBS Dictionary of Engineering Simulator Development for Korea Regional Transport," KARI Technical Report, KARI-TR-RT-97007.

[2] VMEexec System Concepts Manual, Release 4.0, Motorola, Inc .

[3] Airplane Simulator Qualification AC No. 120-40C, Federal Aviation Administration Advisory Circular, U.S. Department of Transportation.

[4] S.K. Advani, B. van der Geest, A. Wattimury, P.C.A. van Gool, "Experiment Control Station for the SIMONA Research Simulator," AIAA-96-3509-CP from AIAA Flight Simulation Technologies Conference, July 1996.

ACKNOWLEGEMENTS

This project was supported by Ministry of Commerce, Industry and Energy as a part of the "Korea Regional Transport Aircraft Development Program – Test & Evaluation."

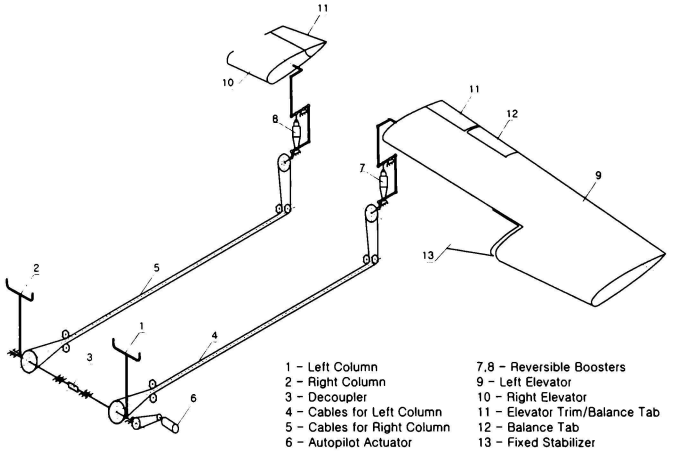


Figure 17. Typical Elevator Control System

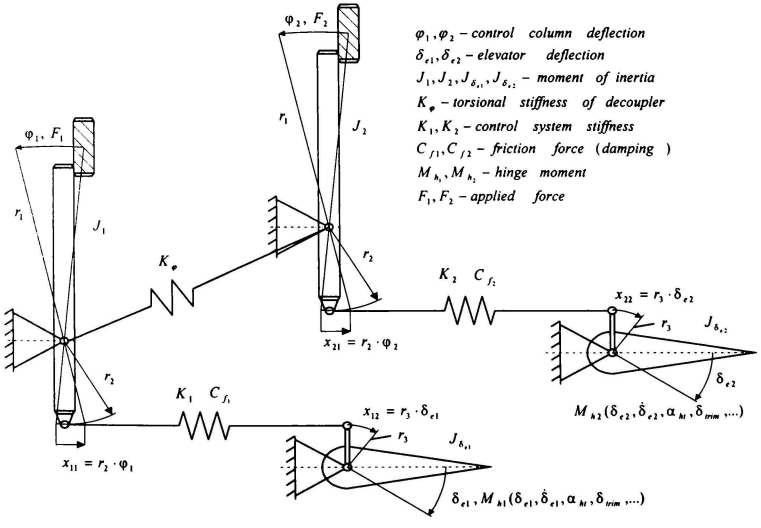


Figure 18. A Hypothetical Model of Elevator Control System

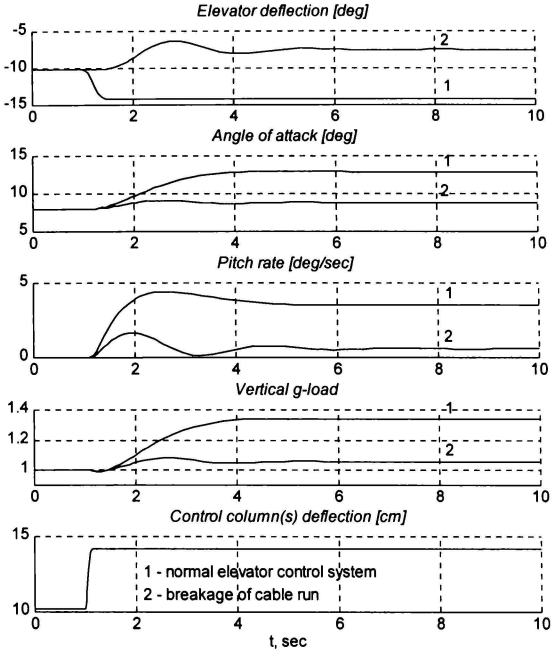


Figure 19. Response to Column Input.

RECENT IMPROVEMENTS TO THE LARGE AMPLITUDE MULTI-MODE AEROSPACE RESEARCH SIMULATOR (LAMARS)

Brian K. Stadler*

Air Force Research Laboratory
Air Vehicles Directorate
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

The Large Amplitude Multi-Mode Aerospace Research Simulator (LAMARS) has been the Air Force Research Laboratory Air Vehicle Directorate's primary handling qualities simulator since the mid 70's. Built by Northrop Corporation for the USAF Flight Dynamics Laboratory, this venerable simulator has gone through a variety of facelifts and improvements that have increased its utility beyond just flying qualities experiments. During the late eighties and early nineties, work was completed on improving the LAMARS visual system which entailed repairs and resurfacing of the spherical projection screen. In addition, new color projectors have been installed to provide increased out-the-window coverage and improved image quality which, when coupled with the motion system, provides an excellent emulation of the flight environment. In order to keep current with advanced weapon system programs, cockpit equipment upgrades have included raster multi-function displays, sound system, and installation of a magnetic head tracker system for helmet mounted display studies. Other visual system upgrades include the incorporation of two laser target projectors for drawing calligraphic images as well as a new low-resolution projector to replace the original sky-earth projector. These upgrades became critical along with better documentation of the motion system as part of in-house experiments in the area of pilot induced oscillations (PIO).

SIMULATION FACILITY

FACILITY OVERVIEW

When originally constructed in the mid-70's, the LAMARS was the primary simulator for the Flight Control Development Laboratory at Wright-Patterson AFB. Since the facility's construction and numerous name changes, the LAMARS is now one of several simulators operated by the Integration and Assessment Branch. In addition to the LAMARS, the Integration and Assessment Branch also operates and maintains manned combat stations for multi-player combat simulations, a 40-ft dome simulator, and a new multi-panel collimated display simulator built by Electro-Visual Engineering (EVE). All of the simulators can be used independently or in conjunction with each other and are integrated with computers located on a controlled centralized computer deck. This implementation permits simulations to share and scale resources based on customer demands. All simulation related computers used for the LAMARS and other facility simulators are housed in a centrally located computer deck as shown below in Figure 1.

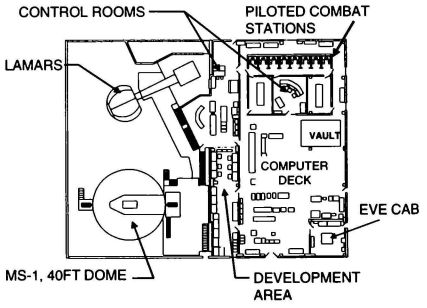


Figure 1, Computer Deck and Simulator High Bay

* Aerospace Engineer, Member AIAA

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States

COMPUTERS

Software for simulating aircraft, weapons, and avionics software as well as software for motion drives and projector drives is executed on various Encore RSX and Encore 91 computers. The RSX machines generally act as the master processor in all real-time simulations and conduct all simulation input/output (I/O). Simulation I/O to other computers or devices is accomplished via Scramnet, Ethernet, or Reflective memory. The Encore RSX machines also house highway drivers that communicate to the KineticSystems Corporation analog-to-digital (A/D), digital-to-analog (D/A), and discrete switch system. Cockpit display symbology for head-down, head-up, or helmet mounted displays (HDD, HUD, HMD respectively) is generated with either an IMI computer for stroke displays or Silicon Graphics computers for raster displays. The facility finally replaced its original terrain boards and now uses two Evans and Sutherland 4530 image generators for out-the-window scene generation. Figure 2 below is a high level schematic of the Integration and Assessment Branch's computer architecture.

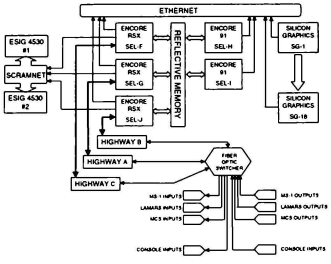


Figure 2, Computer Architecture

I/O SYSTEM

In order to increase simulation facility flexibility for development and operation, the simulation computers and platforms are separated via a programmable I/O system. The primary mechanization for separating the computers from the simulator platforms is via a computer independent I/O system. Facility analog-to-digital, digital-to-analog, discrete, and RS-232 communications to various platform hardware components is accomplished using an Enhanced CAMAC (Computer Automated Measurement and Control IEEE-STD 583) system built by Kinetic Systems Corporation[1]. The CAMAC system consists of a Serial Highway Driver (SHD) and up to 62 remote crates connected by dedicated high-speed fiber optic cables in a ring network. A 'Crate' is a CAMAC card rack and power supply that contains various A/D, D/A, discrete, and RS-232 modules. Each facility simulator platform and control room has its own dedicated CAMAC Crate for I/O. The SHD is the interface between the Encore host computers and the CAMAC system. All data travels serially around the network in one direction.

The enhanced version of CAMAC reduces the communication overhead by providing memory local to the SHD and Crates in which to store commands to execute I/O operations. During the initialization of a simulation, a list of the I/O operations to be performed is written to the SHD and Crates. Then, while the simulation is operating, only data that needs to be transferred during a simulation frame is transferred by the no-wait I/O operation of the High Speed Device (HSD).

Each platform crate holds up to 25 different CAMAC modules allowing for expansion and variation between simulation programs. The D/A modules contain 6 channels with a dedicated 16 bit digital to analog converter. Full-scale range is ±10 volts and output impedance is a maximum of 0.2 ohms capable of driving ±5 milliamperes. The output settling time is a maximum of 10 microseconds and binary data is accepted as a 16 bit two's complement integer. The A/D modules contain 6 channels with a dedicated differential input amplifier and a dedicated 16 bit analog to digital converter. Full scale range is ±10 volts and the input impedence is a minimum of 50 megohms. Each channel has a 3 pole Bessel filter with variable cutoff frequencies with bybass[2]. The converted data is stored as a 16 bit two's complement integer. Typical conversion time is 20-25 microseconds. The discrete input modules have 48 channels each capable of reading one discrete value. Data is read in 24 bit groups into a 24 bit word, each bit representing one channel. The RS-232 module contains 4 independent channels with each having software configurable communications parameters, a DTR/DSR hardware handshake or XON/XOFF handshaking protocol. The receiver and transmitter each have a 1024 character buffer.

During simulation operation, stick and throttle positions are converted via an A/D module located in a particular Crate where they are read by the SHD. The voltages are then retrieved by a HSD located in one of the Encore master computers scaled and placed into Reflective memory for use by simulation software. Discrete values are passed the same way except they use a discrete module in the Crate. Any outputs to the cockpit are written by the HSD into the SHD and then passed to the appropriate Crate D/A or discrete card

which then passes the values to the cockpit hardware. A typical time for reading and converting 12 channels of A/D and reading 48 bits of discrete data is 420μsec.

Three facility Encore RSX computers are connected to separate CAMAC SHDs permitting three separate simulations to be under development or operation at one time. A PC controlled fiber optic switcher is used to setup the various Crates used for a particular SHD and simulation. This permits each SHD to act independently of the others permitting increased facility utilization.

The LAMARS has two dedicated Crates. The first Crate is used exclusively for the motion system due to the fact that there are dedicated accelerometers, gyros, and tachometers in the motion system feedback loops to maintain LAMARS motion performance. However, between the Crate connections and LAMARS hardware is the LAMARS operators' console. Without LAMARS Console consent switches enabled, the simulation is unable to drive the LAMARS' mechanical subsystems. The second LAMARS Crate handles cockpit I/O and servo mechanical target projectors.

LAMARS

When originally built, the LAMARS, Figure 3, was the centerpiece of the new Flight Control Research Laboratory at Wright-Patterson Air Force Base. The new facility housed the LAMARS as well as two other cascade type motion platforms: a multi-crew platform and a fighter-bomber platform. The original visual system for all the simulators in the late 70's and early 80's consisted of two terrain boards. The primary terrain board used for most simulations had 1500:1 scale and was used primarily for approach and take-off simulations. The 1500:1 board was also included as a small portion of the 5000:1 board, which was available for long range and up-and-away tasks.

Originally the LAMARS had an extremely limited visual system. The cockpit out-the-window scenery consisted of a Sky-Earth projector coupled with monochromatic 60x60 degree CRT projector for terrain imagery. The CRT projector imagery originated from the terrain board cameras and was normally fixed forward. There was the potential for using the CRT projector in a head slaved mode since the same CRT projector doubled as the high resolution target projector. However, this option was never used. When air-to-air target tracking tasks were conducted, the CRT projector was used to project a target image provided by a model system that used a wire mounted model enclosed in a camera projection box[3].

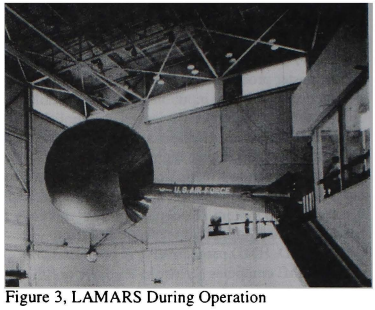


Figure 3, LAMARS During Operation

COCKPIT

The LAMARS cockpit is designed to emulate current and future single place cockpits, Figure 4. The basic structure is designed for easy panel swap out and reconfiguration. The seat can also be configured for straight back or highly inclined positions. Since the LAMARS is a research simulator the priority in configuring the cockpit is placed on installing only those displays required for a particular test in order to represent the vehicle's actual cockpit. This is done in order to keep costs to the customer to a minimum. In the past, the cockpit has also been configured with a 29" monitor and touch screen in order to evaluate advanced pilot vehicle interface (PVI) display concepts. In addition, a full sound system was installed in the cockpit to provide aircraft background noise as well as aircraft system caution, warning, and weapons cueing sounds[4].

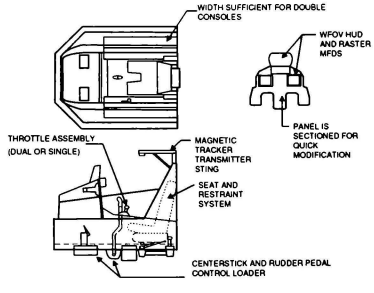


Figure 4, LAMARS Cockpit Configuration

FEEL SYSTEM

Accurate replication of aircraft feel system controls is critical to pilot acceptance of a simulation and important in flying qualities evaluations[5]. The ability to rapidly change force-feel system characteristics for a test or modify a system to reflect changes that may occur during a flight test program are also important. Because of this, the LAMARS cockpit is outfitted with a McFadden variable control loader. The system consists of a centerstick that can accept various grip types and rudder pedals. Both the rudder pedals and the pitch and roll axis of the centerstick have independent electronic controls that permit various breakouts, force gradients, and deadbands to be programmed. The centerstick is attached to an electro-hydraulic actuator assembly that consists of an actuator shaft floating on hydrostatic fluid bearings under pressure to eliminate static friction. Further reductions in friction are accomplished via a sealess housing system that requires the use of a hydraulic scavenging system[6]. Performance features of the LAMARS McFadden system are listed in Table 1.

Performance Parameter	Pitch	Roll	Yaw
Required Travel	7 in. aft 5 in. fwd	±7 in.	±3.25 in.
Electrical Stops	±0.5 in. to full travel	Same	Same
Stop Gradient	80 lb/in min.	Same	Same
Maximum Force	150± 15lb	Same	100 ±10lb
Velocity Limit	60 ±6 in/sec	Same	Same
Spring Gradient[1]	0 to 30 lb/in	Same	0 to 80 lb/in
Breakout Force, min. range	0.25 to 25 lb		0.50 to 50 lb
Friction, min. range	0 to 15 lb	Same	0 to 40 lb
Deadband, min. range	0 to ± 1.0 in	Same	Same
Damping, min. range	0 to 0.5 lb/in/sec	Same	0 to 3.0 lb/in/sec
Trim rate, min. range	0 to 2.0 in/sec	Same	Same
Repeatabiliy	±0.25 lb	±0.20 lb	±0.25 lb

Note 1: Minimum range for each of two straight line segments each side of zero.

Table 1, McFadden Control Loader Characteristics

The LAMARS cockpit can also be outfitted with a variety of throttles. This permits the cockpit to be configured for either single or multi-engine aircraft depending on the test. In addition, the single engine throttle grip can be backdriven by an electric actuator. In order to represent modern fighter aircraft the right side of the cockpit has also been modified for installation of various sidestick force transducer mounts. During simulations requiring the use of a sidestick, the McFadden centerstick is removed and only the pedals are used.

COCKPIT DISPLAYS

When originally delivered, the LAMARS had a simple CRT monitor as a cockpit display. Since then, numerous programs have upgraded the display systems to include various head down and head up displays to increase cockpit flexibility and utility enabling the LAMARS to be configured as a variety of aircraft. The forward cockpit panel is divided into sections allowing for easy reconfiguration and modification. The forward cockpit panel can be configured with a center panel cluster containing mechanical gauges, 5" raster multi-function displays (MFD), as well as various smaller instrument panels. Side panels easily disconnect permitting specialized program support panels to be mounted in the cockpit. Head up display (HUD) symbology is provided by a Kaiser wide field of view HUD. The Kaiser HUD is stroke with a raster inset and is collimated to 10ft in order to overlay the uncollimated out-the-window scenery and has a 15°(vertical) x 28°(horizontal) instantaneous binocular field-of-view (FOV) at the design eye. In addition to the HUD display, the LAMARS is outfitted with a stroke only Kaiser Agile Eye 12° field-of-view helmet mounted display (HMD). HMD position and orientation information is generated via a Polhemus Fastrack magnetic tracker system with a 19" (wide) x 14" (tall) x 17" (deep) motion box centered about the pilot's eyepoint. The tracker sensor is externally mounted to the HMD permitting its use with other HMD systems or for recording pilot head position and orientation when mounted to a regular helmet. Due to the mix of raster and stroke cockpit displays, the facility uses Silicon Graphics computers for raster displays while a multi-channel IMI stroke graphics machine is maintained for developing and generating stroke displays for the HUD and HMD. The LAMARS forward panels have also been removed and a 29" CRT with touch screen installed to provide a glass cockpit capability.

VISUAL

The LAMARS visual system has gone through the most changes and improvements over the past 10 years. The visual system upgrades have not only improved the utility of the LAMARS for static testing but have enhanced the pilot's experience when flying the LAMARS with motion. The LAMARS visual display system consists of a 20ft diameter spherical projection screen surrounding a single place cockpit and mounted to the cockpit-sphere gimbal system, Figure 5. The projection sphere is designed such that the pilot's eyes

are at the exact center of the sphere. The cockpit gimbaling system and projector mounts are directly behind the cockpit and limit the total field-of-regard (FOR) of the projection screen. Because of this, the screen FOR is limited to 266° in azimuth and 106° in upward elevation from the pilot's eyepoint. The sphere projection surface was resurfaced and painted in August of 1992 in order to fix blemishes and minor damage that had accumulated over the years. The sphere screen gain was measured to be 0.7 after resurfacing and painting. LAMARS out-the-window scenery has been markedly improved by the installation of five Sharp XG-650UB color liquid crystal display (LCD) projectors mounted directly behind the cockpit[7]. Five LCD projectors provide a total field of view of approximately 120° x 40° which is a vast improvement over the original system's dual monochromatic 60° x 60° FOV high-resolution projector and sky-earth projector. An approximation is provided for the total field of view because it varies in azimuth due to the projector mounting and orientation. Clarification of the LAMARS FOV is provided in Figure 6 which shows an Aitoff projection of the various projector mappings in relation to the pilot's eyepoint.

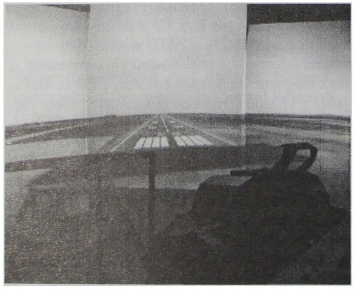


Figure 5, Inside of LAMARS Sphere

Specifications for the LCD projectors and screen characteristics are provided in Table 2. Brightness readings for the LAMARS projectors were taken with all projectors displaying the same test pattern simultaneously. The test pattern consisted of a black background with a white window inset such that 50% of the projection surface was white and the other 50% black. Measurements were taken with all projectors displaying the test pattern simultaneously due to the integrating nature of the spherical projection surface, which tends to wash out video images. With just the center projector in operation, brightness values of 3.02 ft-lamb and contrast ratios of 42.14 (USAF definition) and 43.14 (optical definition) were obtained. Currently, a sixth LCD projector with a special fish eye lens is being developed for use as a 180° x 90° FOV low-resolution background projector.

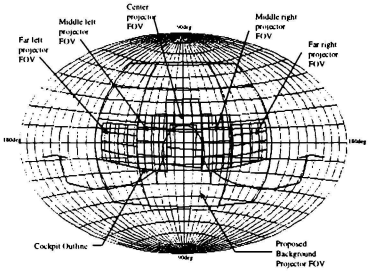


Figure 6, Visual System Projector FOV Regions

Sharp LCD Projector Model XG-E650UB Characteristic			
Display Method	Translucent TN liquid crystal panels		
No. of Pixels	307,200 pixels (640W x 480V)		
Lens	1-1.3 zoom, F5.9-6.9, f=210-270mm		
Contrast Ratio	100:1		
Horizontal Resolution	500 TV lines (video input)		
Drive Method	Thin Film Transistor Active Matrix Panel		
LAMARS Projector Station Screen Gain = 0.7	Brightness	Contrast Ratio (USAF)	Contrast Ratio (Optical)
Far Right	3.65 ft-lamb	15.59	16.59
Middle Right	3.40 ft-lamb	14.45	15.45
Center	3.12 ft-lamb	12.00	13.00
Middle Left	3.82 ft-lamb	15.60	16.60
Far Left	4.08 ft-lamb	18.43	19.43

Table 2, LAMARS Visual Display System Specifications

In addition to the LCD projectors, three target projectors are also mounted behind the cockpit for displaying other aircraft and missiles. One of the projectors is the original dual-mode monochromatic target projector delivered with the LAMARS. This projector originally was used as either a high-resolution target projector with a 15° FOV or switched over for displaying terrain scenery for landing and low-level tasks with a 60° x 60° FOV. In either mode the projector was used in conjunction with a sky-earth projector that displayed a horizon on the screen and moved in relation to aircraft pitch, roll, and yaw rates.

The sky-earth projector has since been removed to make room for the new low resolution background projector and the target projector is now strictly used for displaying targets. A Silicon Graphics machine provides the imagery for the monochromatic target projector, which has a variable image size capable of drawing a target aircraft in the range of 400 feet to 47,600 feet. In addition to the monochromatic target projector, two laser projectors were installed in the LAMARS in 1991. The lasers draw calligraphic targets and are used for drawing both aircraft and missiles during flyout. Specifications for the laser target projectors are listed in Table 3.

Laser Type	594.1 nm HeNe
Brightness	1.5 ft-lamb. Min. displaying on a 5"X16" uniform white patch
Image Size	1 arcmin spot focused at a 240" projection distance to a 5.5 deg. Max image size
Useful Simulated Range	500' to 60,000' for an aircraft of 50' length and 33' wingspan at 240" projection and viewing distance
Servo Velocity	10 rad/sec azimuth and elevation

Table 3, Laser Target Projector Specifications

MOTION SYSTEM

The LAMARS, Figure 7, is referred to as a beam type simulator and is based on a prototype called the Large Amplitude Simulator (LAS) which, until recently, was operated and maintained by Northrop Corp. The motion system consists of a 30-foot long beam driven by hydraulic actuators to provide +/- 10 feet of vertical travel and +/- 10 feet of lateral travel at the cockpit. The cockpit and projection sphere structure is mounted at one end on a gimbal system that provides three degrees of rotational freedom (+/- 25 degrees in pitch, roll, and yaw). The beam provides a rigid low inertia structure which is compatible with minimizing power and control problems associated with high inertia systems[8]. All electrical cabling and hydraulic tubes are routed through the beam to the cockpit and visual display subsystems. System limits, as specified in the original LAMARS Statement of Work, are listed in Table 4[9].

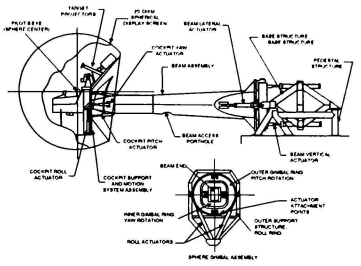


Figure 7, LAMARS

SERVO	TRAVEL	N0-LOAD VELOCITY	ACCELERATION
Pitch Beam θ_B	±10 ft	13 ft/sec	±3 g
Yaw Beam ψ_B		10 ft/sec	±2 g
Pitch Cab θ_C	±25 deg	60 deg/sec	±400 deg/sec^2
Yaw Cab ψ_C			±200 deg/sec^2
Roll Cab ϕ_C			±460 deg/sec^2

Table 4, LAMARS Servo Operational Limits

SAFETY AND SUPPORT EQUIPMENT

Several safety features have been built into the hydraulic system in order to prevent damaging the system in case of emergencies such as power outages. Safety features include a hydraulic counterbalance system in beam vertical motion and remotely operated lockout devices for cockpit pitch and roll actuators. A system of limit switches, combined with internal snubbers, in all cockpit and beam actuators is used to provide a controlled and limited deceleration at the end of each actuator's stroke. Mechanical "lockouts" are provided for all the motion and display system drive mechanisms except in those cases where the static position is at a travel stop. Use of close tolerance rig pins or locks permits repeatable positioning of the motion system permitting their use for nulling feedback potentiometers. A pin is used to prevent cockpit roll motion during pilot ingress and egress into the cockpit sphere while a latch restrains the pitch actuator. These lockouts are pneumatically actuated by remote control from the main monitoring control console.

TRAVEL LIMIT & DECELERATION DEVICES

Safe deceleration and stopping of the motion system is critical and the LAMARS was built with a rather unique system. The following system descriptions are from Reference 3 and provided here for completeness. Desirable stopping characteristics of cam operated deceleration valves are integrated with the actuating cylinder assemblies thereby eliminating attendant interconnecting lines and mechanical linkages. This arrangement is displayed for the vertical axis beam servo in Figure 8. Stopping is initiated near the ends of cylinder travel by straight, close tolerance plugs which enter recesses in the cylinder end caps. These plugs close the path of normal exit flow from the cylinders and force the exit fluid through the only remaining path which is though the pressure relief valves. Since the pressure relief valves are pressure referenced to the opposite side of the pistons, the stopping force is controlled independent of the active state of the servo valve. For stopping which originates at maximum velocity, stopping distances are provided such that relief valve settings limit the decelerations to levels that are no higher than the peak acceleration levels of each servo. These stopping distances are in addition to the useable travels of the motion system listed in Table 4. Due to speed-of-response considerations and reliability, the use of the electrically operated main shutoff valves preclude their use as the primary stopping method.

Assurance against excessive downward velocity of the beam under a complete and sudden failure of the vertical beam servo is a critical safety function of the hydraulically charged beam counterbalance system. The method of hydraulic counterbalance is included in Figure 8. The counterbalance actuators are the forward sections of the tandem main cylinders of the vertical beam servo. To balance the beam at mid travel, the hydraulic circuit of the counterbalance is charged from the central-source supply to raise pneumatic accumulator pressure to 775psi, or approximately 6-gallons of accumulator fluid charge. With the accumulator fill and drain valves closed, full travel of the beam corresponds to variation in the counterbalance pressure from 650psi (beam full up) to 925psi (beam full down). The counterbalance fill and drain rates are controlled by fixed orifices to preclude any sudden change in the amount of the trapped fluid.

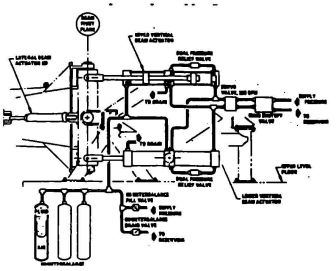


Figure 8, Hydraulic System

HYDRAULIC SUBSYSTEM

Hydraulic actuators are used in all five axes of the motion system to provide precise position control of the beam and cockpit sphere. Two cylinders operate in a "push-pull" arrangement on each axes of motion. The hydraulic power supply and distribution system is designed to operate at 2700psi pressure. The hydraulic pump provides 100 gallons per minute (GPM) of hydraulic flow rate and has recently been rebuilt to increase its life. The hydraulic pump is driven by a 200HP electric motor. A self-contained reservoir below the LAMARS pedestal holds 25 gallons of hydraulic oil. The vertical beam control valve is a series 79 Moog servo valve rated at 200 GPM while the control valves on the other axes are Series 72 Moog industrial valves. Three 10 gallon accumulators assist the pump supply during maximum velocity commands. Over pressure relief is provided by two relief valves across the ports of each cylinder in case the servo valves suddenly close.

SERVO PERFORMANCE

The ultimate goal of the LAMARS motion system is to generate excellent acceleration cues to the pilot while performing various tasks. In order to accomplish this, the LAMARS motion system has a pair of hydraulic actuators for each of the platform's five degrees of freedom. Three of the servo pairs are used to generate pitch, roll, and yaw angular motions of the cockpit. The remaining two pairs of servos are used to generate beam heave and sway motions. Motion system performance criteria is based on replicating acceleration cues at the pilot station as accurately and smoothly as possible. The LAMARS motion system is designed to maintain roughness in overall acceleration below .06g in the translational servos and below 15deg/sec2 in the rotational servos[8]. Since the LAMARS is a beam type simulator, there is a major concern for generating spurious acceleration signals at undesirable frequencies

other than those commanded due to beam structural bending modes. Elimination of structural modes in all axes is achieved by the use of acceleration feedback in addition to velocity and position feedback in the beam servo drives[10]. In order to maintain synchronization between the visual servo mechanical projectors and the motion system, all the LAMARS' servos are designed to match a 0.7 damped 25 rad/sec second order system referenced at the pilot station. This is achieved using position, velocity and acceleration feedback in the servo electronics and in software to remove dynamics associated with the electronic compensation.

Figures 9 through 18 provided at the end of the paper are plots of the LAMARS motion platform excursion limits and frequency response plots of the various beam servo axes. The excursion plots are log-log plots showing the system limits and operational limits of the LAMARS. The straight lines represent the position, velocity, and acceleration limits tabulated in Table 4 above. The far-left diagonal line is the position limit, the vertical line at the top of each plot is the velocity limit, and the far-right diagonal line represents acceleration limit of each servo axes. The points on each excursion plot represent actual system measurements by inputting a sine wave at the given frequency and represent the upper limit of what the platform system can generate before either a g-limit or system limit is reached. The frequency response plots compare each LAMARS servo axis uncompensated system to the ideal compensated system. A sum-of-sines signal is input into each servo axis system and then a model is fitted to the input/output data in order to generate the data for the uncompensated system. This procedure is similar to that recommended in AGARD-AR-144 'Dynamic Characteristics of Flight Simulator Motion Systems"[11]. However, the frequency plots shown are for the position response of the servo axes not acceleration. Further work on the LAMARS is planned to obtain the system characteristic plots recommended in reference 11 for comparing different motion platforms. Further data is to be gathered including acceleration response describing functions, linearity and noise, hystersis, and dynamic threshold measurements.

MOTION DRIVES

The facility still maintains and uses the original motion drive logic delivered with the LAMARS. The motion washout logic was developed by John Sinacori, then an engineer at Northrop, and has been used successfully on the LAMARS for some time[12]. The motion washout logic uses aircraft angular rates and linear accelerations as inputs. The angular rates are sent through first order washouts, then transformed into inertial coordinates, before passing through a final integrator. The linear accelerations of the aircraft are first passed through a series of filters, gravity compensators, and washouts before final integration. Once the required acceleration, velocity, and position terms are calculated, cockpit leveling is done before final filtering of the drive signals through the platform compensation equations. After this, the signals are sent to the D/A converters to drive the platform.

Work is currently underway investigating the possible use of non-linear adaptive motion drives on the LAMARS. An initial attempt was started as part of research into improving the amount of motion response obtained beyond the current washout scheme. This research was started in order to improve vertical acceleration cues due to experiments in pilot induced oscillations in aircraft. Initial results have shown promise in obtaining more useable travel out of the motion system. However, work still needs to be done in several areas before the non-linear drives can replace the existing linear drives.

SUMMARY

The LAMARS is one of a handful of large amplitude motion simulators still remaining in operation today. In order to maintain its utility beyond just flying qualities experiments, improvements in cockpit, visual, and sound systems have been accomplished increasing the simulator's flexibility and cost effectiveness. Work is continuing to improve the visual system and motion system in order to keep the system current with new technology and capable of simulating aircraft into the next century.

REFERENCES

1. Mallery, Craig, *Computer Automated Measurement and Control Optimized for R&D Aircraft Simulation*, Master's Thesis, Wright State University, 1992

2. Anon., *Model 3595-A2A, E2A 6-Channel, 16 Bit A/D Convertor Instruction Manual*, KineticSystems Corp., 1988

3. Haas, R. L., Hotz, H.E., and Mills, G. R., "The Large Amplitude Multi-Mode Aerospace Research (LAMAR) Simulator", AIAA-73-922, 1973

4. Sturgell, Mark E., and Hebert, Jeffery, "Applying Digital Music Instrument Technology to Air-Combat Simulation", AIAA-92-4164-CP, 1992

5. McFadden, B.W., Joas, John G., "Accurately Reproducing Pilot's Control Forces in a Flight Simulation", AIAA-78-1586, 1978

6. Anon., *Instruction Manual Model 392A8 Variable Cockpit Control Loading System*, McFadden Electronics Co.

7. Anon., *XG-E650UB Operation Manua*", Sharp Electronics Corporation

8. Anon., *Operating & Maintenance Manual for the Large Amplitude Aeronautical Research Simulator (LAMARS)*, NOR75-5, Northrop Corp., Contract F33615-72-C-0463

9. Mills, G.R., "ITWO 2406 Motion System Servo Performance Test", Unpublished Memorandum,.. 1975

10. Mill, G.R., "Improving the Quality of Motion Representation in Moving Base, Piloted Flight Simulators", *Journal of Aircraft*, Vol. 4 No. 5, Sept-Oct 1967

11. Lean. D.,, "Dynamic Characteristics of Flight Simulator Motion Systems", AGARD-AR-144, 1979

12. Sinacori, J. B., "A Practical Approach to Motion Simulation", AIAA 73-931, 1973

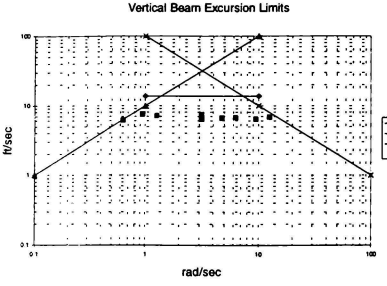


Figure 9, Vertical Beam Excursion Limits

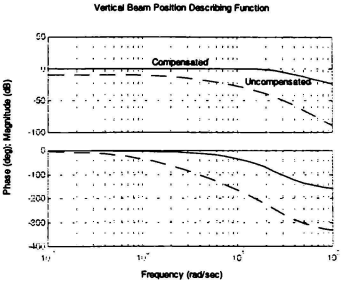


Figure 10, Vertical Beam Frequency Response

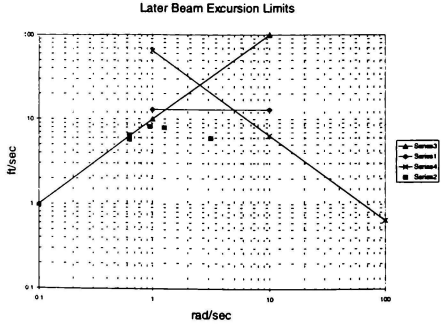


Figure 11, Lateral Beam Excursion Limits

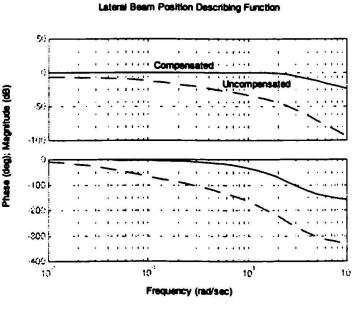


Figure 12, Lateral Beam Frequency Response

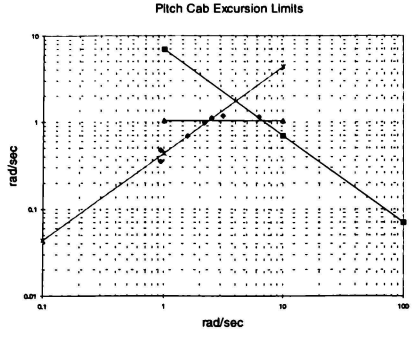


Figure 13, Pitch Cab Excursion Limits

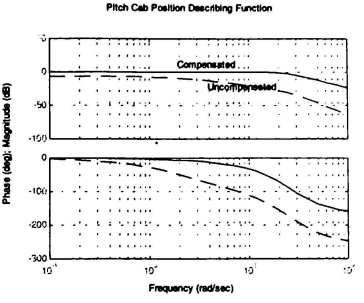


Figure 14, Pitch Cab Frequency Response

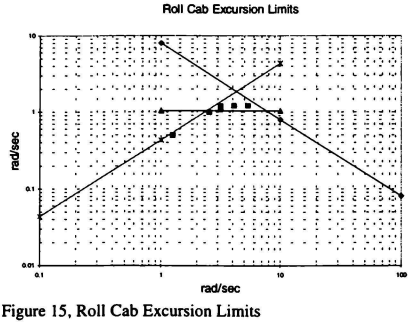


Figure 15, Roll Cab Excursion Limits

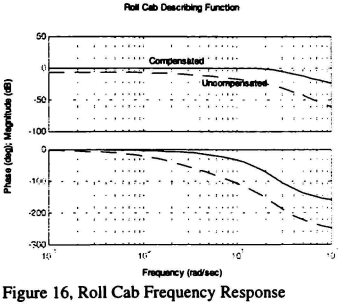


Figure 16, Roll Cab Frequency Response

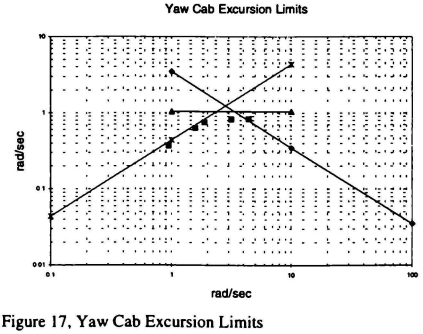


Figure 17, Yaw Cab Excursion Limits

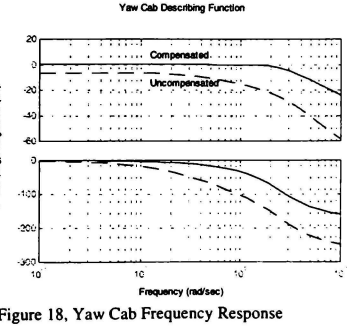


Figure 18, Yaw Cab Frequency Response

INFLUENCE OF POSITION SENSOR PROPERTIES WHEN USING A PERSPECTIVE FLIGHTPATH DISPLAY

Dr. E. Theunissen and R.E. de Bruijn, Department of Electrical Engineering, Delft University of Technology, The Netherlands, e.theunissen@et.tudelft.nl*

ABSTRACT

Data latency, data update rate, error spectrum and display update rate are discussed in the context of control- and perceptual requirements. Examples of applications are provided in which the limited update rate must be compensated for, and the concept of extrapolation with data smoothing is identified as a potential solution to meet the perceptual- and control-based requirements. A requirements analysis is performed for the position data driving a perspective flightpath display, and a so-called datashaper is discussed that is derived from an extrapolation/smoothing concept originally developed for distributed interactive simulation applications. To deal with sudden changes in the position error estimate, the concept is expanded with a threshold predictor. To quantify the performance of the datashaper, an evaluation has been performed.

INTRODUCTION

Various studies have demonstrated the feasibility and the benefits of perspective flightpath displays for accurate manual guidance and control purposes both in simulator evaluations[6,12,14] and in real flight[11,13]. The depiction of the perspective flightpath requires 3-D position data. To obtain a smoothly animated display, the update rate of the data used to create a snapshot of the situation must exceed 15 Hz. In case the data rate from the positioning system does not satisfy this requirement, extrapolation must be used. This paper discusses an approach that can be used to generate a smoothly animated display and to reduce the likelihood of position estimation error (PEE) control motion noise by means of an extrapolation and smoothing method that can be implemented in the display electronics unit (DEU).

BACKGROUND OF THE PROBLEM

A smoothly animated display poses requirements on the data determining the contents of the display in terms of update rate, resolution, and noise spectrum. When the display is used for manual control, the stability of the closed-loop system determines the system latency requirements.
Due to a variety of reasons, the data that is available may not meet all the requirements to obtain a smoothly animated display. This can be due to limitations in bandwidth of the interface between the sensor(s) and the display system, an error spectrum of the sensor that has some undesirable components, or a too large processing delay causing an unacceptable system latency.

System latency

Funk et al.[4] define avionics system latency as '*the time delay between aircraft motion and the corresponding indication of that motion on the aircraft displays*'. System latency can be subdivided into the data measurement latency, the data transmission latency, and the data presentation latency (Fig. 1). The data measurement latency is determined by the sensor properties. The data transmission latency is determined by the type of interface between the measurement system and the data presentation system. The data presentation latency is determined by the maximum display update rate and the scheduling* used in the DEU.
When multiple instruments are depicted on a single electronic display, as for example is the case with the primary flight display (PFD), each instrument can have a different system latency.

---

*Member AIAA

Copyright © 1999 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission

*Scheduling can be used to achieve a sufficiently high display update rate within the performance constraints imposed by the display processor(s). With several of today's primary flight displays for example, the artificial horizon and the pitch attitude tape are presented at a higher update rate than other symbology.

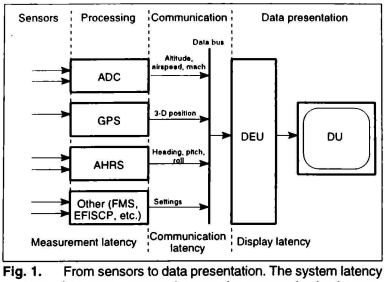


Fig. 1. From sensors to data presentation. The system latency consists of the measurement latency, the communication latency, and the display latency.

The system latency of the attitude data has the biggest influence on the perception of handling qualities. In AC 25-11[1] the requirements with respect to display data latency are stated as: '*Any lag introduced by the display system should be consistent with the airplane control task associated with that parameter. In particular, display system lag (including the sensor) for attitude should not exceed a first order equivalent time constant of 100 milliseconds for airplanes with conventional control system response*'.

Data update rate

To depict the flightpath as seen from the aircraft for guidance and control purposes, snapshots of the flightpath as seen from the actual position and orientation must be generated. The successive presentation of images showing a perspective projection of the 3-D flightpath yields an optic flow pattern that can be described by velocity vectors (Fig. 2).

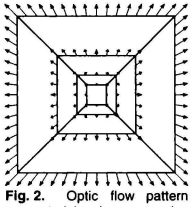


Fig. 2. Optic flow pattern generated by the successive presentation of perspective images of the flightpath.

To convey an illusion of continuous motion, the time between the successive presentation of images must be sufficiently small. The illusion of continuous motion does not only depend on the ability of the display hardware to generate images at a sufficiently high update rate, but also on the update rate of the data needed to generate the image. In AC 25-11[1] the perceptual requirements with respect to display data update rate are stated as: '*For those elements of the display that are normally in motion, any jitter, jerkiness, or ratcheting effect should neither be distracting nor objectionable. Screen data update rates for analog symbols used in direct airplane or powerplant manual control tasks (such as attitude, engine parameters, etc.) should be equal to or greater than 15 Hz*'.

The maximum required update rate for each displayed element is determined by the scaling that is applied to the data that drives the transformations of that element. When expressing the rate of change of the variable that drives a certain display element in minimum discernable display units per second, the maximum update rate that still yields a change between each successive image can be calculated. For example, when the resolution of a navigation display is 2048 pixels, with a visible range of 20 miles and an aircraft velocity of 220 kts, the motion of the elements on the navigation display is approximately 6.26 pixels/sec. An update rate that is higher than 6.26 Hz will yield less than a pixel displacement between successive images.

Position error spectrum

Sudden changes in the position, for example caused by a step change in the PEE, can cause a perception of discontinuities in the optic flow pattern, which in turn can result in discomfort, a reduction of confidence, a control action, or draw unnecessary attention. Therefore, perceptual requirements should also address the likelihood of discontinuities in the optic flow pattern.

EXAMPLES OF THE PROBLEM

Since the problem does not necessarily apply to all data used in the generation of an image, the way it manifests itself depends on the specific implementation. Two examples in which the limited position data update rate poses a problem are distributed interactive simulation (DIS) applications, and navigation/guidance displays that have a sufficiently high position resolution.

DIS applications

With DIS applications, it is typically the limited bandwidth of the network connecting the participating simulators that restricts the data update rate. As a result, all data regarding other vehicles in the simulation suffers from the limitations in update rate and the resulting latency. Johns[7] describes a study into the effects of delay times in a dome-to-dome simulator link for air-to-air combat, and the use of data age adjustment to compensate for adverse delay effects. Katz and Graham[8] describe an evaluation of various algorithms for the extrapolation of airplane states. One problem with extrapolation is that once a new update becomes available, the error between the extrapolated position and the new update may lead to jumps in the generated image at the frequency of the sensor data update rate. Lin et al.[9] discuss two algorithms for the smoothing of a dead-reckoned image in distributed interactive simulation (DIS).

Guidance and navigation displays

With respect to guidance and navigation displays, Battiste et al.[2] describe the evaluation of a ground taxi map display for the support of low visibility surface operations. Their display shows an airport layout with the aircraft's current position, taxiway center lines, other traffic, and ATCS clearance information. The aim of the display was '*to provide integrated vehicle control, navigation, and operational situation information in an intuitive and quickly identifiable and understandable manner*'. The position data update rate of the display was limited to 2 Hz, and they report an increase in the disparity of the aircraft position with increased speed that they attribute to this limited update rate. They also indicate that subjects commented that the '*position display could update more quickly*'.

POTENTIAL SOLUTIONS AND CONSTRAINTS

A potential solution can be achieved either by removing the bottlenecks that cause the problem or by implementing methods that compensate for the problem caused by the bottlenecks. Viable solutions are determined by the effort needed to remove the cause of the bottleneck and the effort needed to sufficiently compensate for the undesired effects caused by the bottleneck.

Current approaches

With respect to the earlier mentioned airport surface map display, Battiste et al.[2] propose to integrate DGPS data with the inertial navigation system (INS) on the aircraft. This approach has been used by Sachs and Möller[11] for their flight trials with a synthetic vision display that included a perspective flightpath. Fig. 3 shows an overview of such an approach.

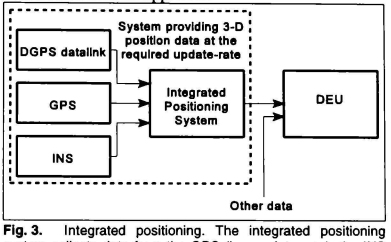


Fig. 3. Integrated positioning. The integrated positioning system collects data from the GPS (low update rate), the INS (high update rate), and obtains differential GPS corrections through a DGPS datalink. It outputs the position data at the update rate needed to achieve a smoothly animated display.

However, integrating DGPS position data with an INS is not the only solution. For the flight trials performed in 1994 by Delft University, a Kalman predictor was used to provide position estimates at an update rate of 20 Hz. Fig. 4 shows an overview of this approach.

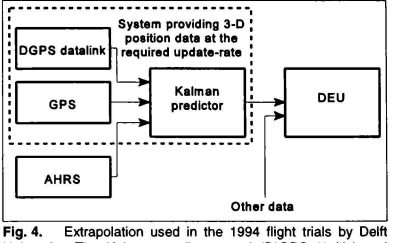


Fig. 4. Extrapolation used in the 1994 flight trials by Delft University. The Kalman predictor used (D)GPS (1 Hz) and attitude data (20 Hz) to extrapolate the position of the aircraft. A dedicated interface provided the DEU with the position data at the desired update-rate.

Alternatives

In both approaches a dedicated interface to the DEU was used to provide the DEU with all data at the required display update rate.

At present, several research programs address potential improvements in the flight instruments of general aviation (GA) aircraft, one of which is AGATE[3]. The concept of the perspective flightpath display has frequently been mentioned as a potential improvement. It is unlikely that the next generation of GA aircraft will be equipped with an INS. It is much more likely that they will be equipped with GPS receivers, and perhaps with a DGPS/WAAS capability. The system in such aircraft will not have a positioning system specifically suited to the requirements of the 3-D data presentation. Therefore, it is deemed realistic to assume that the DEU will be connected to an on-board databus that provides the DEU with position data at an update rate that is determined by the receiver characteristics (Fig. 5).

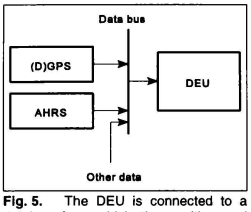


Fig. 5. The DEU is connected to a databus from which the position and attitude data is obtained. Typically, position data will be available at a maximum rate of 5 Hz.

The minimum update rate and the maximum latency that are required, are defined in the Minimum Operational Performance Standards (MOPS) for GPS/WAAS Equipment[10].

Position sensor requirements

In RTCA DO-229[10], the GPS receiver update rate and latency requirements for en route and terminal mode are specified as follows:

- *'The minimum update rate of position outputs used for navigation shall be once per second'.*
- *'The latency of position output, defined as the interval between the time of measurement and the time of applicability of the position, shall be less than or equal to 500 milliseconds. The data defining the position shall be output prior to 200 milliseconds after the time of applicability'.*

The update rate and latency requirements for non-precision approach mode are equivalent to the update rate and latency requirements for en route and terminal mode. For precision approach operations the requirements for update rate and latency are specified as:

- *'The GPS/WAAS equipment shall output updated position data at a minimum of 5 times per second'.*
- *'The latency of the position output, defined as the interval between the time of measurement and the time of applicability of the position, shall be less than or equal to 200 milliseconds. The data defining the position shall be output prior to 100 milliseconds after the time of applicability'.*

It is also stated that *'some autopilots may require a higher update rate'.*

If in these airplanes display concepts are employed that rely on 3-D data presentation, the most efficient way to obtain the required position data update rate for the display is to use extrapolation (dead-reckoning) in the DEU. Additional measures such as data smoothing need to be taken to prevent disruptions in the optic flow pattern. To define the requirements of the extrapolation and smoothing process, an analysis of the transformation of position data into visual cues is needed. Based on such an analysis, assumptions regarding the influence of the type and magnitude of undesired visual cues on the control strategy can be made. From these assumptions, guidelines for the initial selection of values determining the representation of the flightpath and the parameters determining the smoothing performed by the datashaper can be derived. It is evident that this will be an iterative process.

REQUIREMENTS ANALYSIS

With a perspective flightpath display, the representation of the desired flightpath as seen from the aircraft evokes holistic perception. The orientation and features of the object contain the information about the roll error, track angle error, flight path angle error, cross track error, and vertical track error. The design parameters of the perspective flightpath display (tunnel width, tunnel height, horizontal field of view, vertical field of view and minimum viewing distance) determine the magnitude of the various types of visual cues. Theunissen[13] describes how position and orientation errors of the aircraft determine the transformations that distort the 2-D symmetry of the perspectively presented flightpath. He also shows how these can be approximated by a splay

angle change, a translation, and a rotation. Fig. 6. provides an overview of the transformations that are applied.

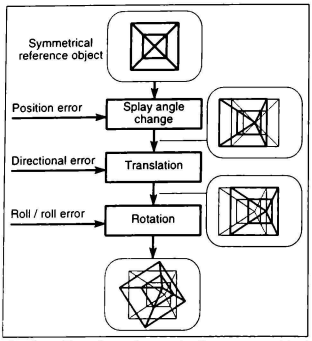


Fig. 6. Transformation of the symmetrical reference shape as a function of position and orientation errors.

The main effect of a change in lateral or vertical position relative to the flightpath, is a change in the splay angle** and a translation of the cross-section frames that is inversely proportional to the distance from the viewpoint. The ratio of the change in the splay angle and the corresponding change in position, is the position error gain. The position error gain and the design parameters of the datashaper should be selected in such a way that the visual cues resulting from a disruption in the optic flow pattern do not lead to a pilot control action. To directly quantify the requirements of the datashaper, a model is needed that predicts pilot control actions as a function of the type and magnitude of the visual cues conveyed by a perspective flightpath display. What is currently unknown is the maximum change in visual cues that does not evoke a control action by the pilot. Although studies showed the relation between control activity and position error gain[12], experiments have also shown that the decision to initiate a control action is not based on separate thresholds for orientation and position errors, but on a threshold that is determined by a combination of the position and orientation errors[13]. The derivation of a model that allows a prediction of the moment and magnitude of a pilot control action as a function of the design parameters, is complicated by the fact that in contrast to the continuous compensatory control strategy used with a flight director, anticipatory and error-neglecting control strategies are possible[13]. To the authors' knowledge, at present no validated pilot model is available that can be used to select a maximum position error gain based on control motion noise restrictions. As a result, when performing an initial experiment to test the datashaper, an educated guess will be required with respect to the initial values of the tunnel size.

EXTRAPOLATION AND SMOOTHING

The structure of the datashaper presented in Fig. 7 is the same as the one used by Lin et al.[9].

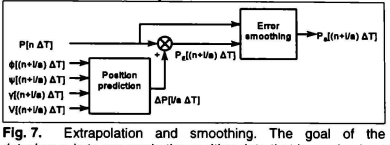


Fig. 7. Extrapolation and smoothing. The goal of the datashaper is to upsample the position data that is coming in at an update rate of f_{POS} to the desired data update rate of the display, f_{DISP}. The ratio between f_{DISP} and f_{POS} is the upsample-ratio *a*. *P[n ΔT]* represents the 3-D position estimate that is used between the time $n\Delta T$ and $(n+1)\Delta T$, with $\Delta T=1/f_{POS}$. A position predictor is run at a frequency f_{DISP}, and the result $\Delta P[i/a\ \Delta T]$ is added to the position *P[n ΔT]*. This yields a position estimate $P_E[(n+i/a)\ \Delta T]$ at the time $(n+i/a)\ \Delta T$, with $0<i<a$. When a new position update $P[(n+1)\ \Delta T]$ becomes available at $n+1$ ($i=a$), there will generally be a difference ΔP_E between the prediction $P_E[(n+(a-1)/a)\ \Delta T]$ and the new update that is larger than the difference $\Delta P[1/a\ \Delta T]$ between two successive predictions. The difference ΔP_E consists of the change in position estimation error and the prediction error. This might cause a disruption in the optic flow pattern at the position data update rate. To prevent this, the error smoother distributes ΔP_E over the following *a* predictions.

It is assumed that the position estimate $P[n\Delta T]$ that becomes available at $t[n]$, is the best estimate, and the history $t[n-1]$, $t[n-2]$,... has been taken into account, e.g. by means of a Kalman filter. To retain a smoothly animated picture when a new sample of the position data becomes available, a correction of the position resulting from the dead-reckoning algorithm will typically be required. To prevent a visual artefact from a too large correction, the difference ΔP_E between the current position estimate P at $t[n]$ and the predicted position P_E based on the best estimate at $t[n-1]$ and the dead-reckoning algorithm over a time ΔT must be gradually added during the next dead-reckoning cycles in such a way that the error component ΔP_E is zero at the time $t[n+1]$ when a new position update becomes available.

** The splay angle is the line between an along track tunnel line and the line perpendicular to the horizon[13]

Influence of the position estimation error

A main difference between the current application and DIS applications, is that a position update in DIS yields the true position, whereas the position update provided by the positioning sensor is an estimate. The difference between the true aircraft position and the position estimated by the positioning system is the position estimation error (PEE). Fig. 8 illustrates the various error components in more detail.

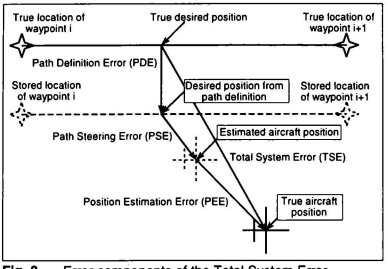


Fig. 8. Error components of the Total System Error.

The error component with DIS applications only comprises the prediction error, but the error component in the application discussed in this paper consists of the prediction error and the change in PEE.

The accuracy of the positioning system is typically quantified by the 95% confidence interval. To prevent control motion noise, one must ensure that a change in the PEE does not result in a pilot control action. For GPS, the change in the PEE can be divided in a low-frequency part, which is mainly caused by the selective availability (SA)[5], and the occurrence of step changes when the set of the satellites used for position determination changes.

Using the position error gain to deal with SA

The visual cues and their change in time determine the pilot's control actions. To minimize control actions induced by the PEE, attention needs to be paid to both the static and to the dynamic visual cues. Therefore, the temporal correlation between the subsequent samples of a PEE must be taken into account.

To select the position error gain so that SA noise does not enter the control loop, data about the maximum change in visual cues that does not evoke a control action, is needed. As mentioned earlier, at present no validated pilot models for perspective displays are available that can be used in this respect. In the absence of such a model, the following assumptions are made:

1. *If the rate of change of the PEE exceeds a certain threshold, the control action is based on the perception of the rate of change.*
2. *If the rate of change of the PEE is below the earlier mentioned threshold, at some point in time the total error may become sufficiently large that a snapshot of the situation is sufficient for the pilot to decide to intervene.*

The position error gain should be selected low enough that the SA causes a rate of change that remains below the earlier mentioned threshold.

Dealing with step changes

When selecting the tunnel size based on the requirement that a step equal to the difference between the 2.5% and 97.5% boundaries of the confidence interval of the PEE does not result in a perceivable change in the presentation, it will lead to very large tunnels for unaugmented GPS. This leads to conflicting requirements with respect to the magnitude of the temporal range cues[13]. If it is possible to smoothen the effects of a step in time in such a way that it will not result in visual cues that lead to a control action, it is not necessary to select the position error gain equal to the range of the 95% confidence interval. In case of a sudden change in PEE. the conventional smoothing may not be sufficient. Fig. 9 illustrates such a situation.

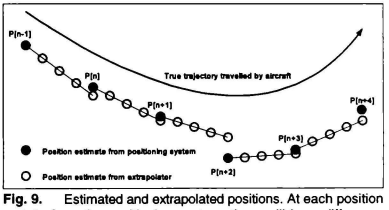


Fig. 9. Estimated and extrapolated positions. At each position update from the positioning system, there will be a difference between the estimated and the extrapolated position. Between *n-1* and *n+1* the change in position estimation error is quite small. At *P[n+2]* there is a change in the position estimation error.

A step change as depicted at *P[n+2]* in Fig. 9 can cause a too steep gradient of the correction, which, given a sufficiently high position error gain, is likely to be noticed

by the pilot and may lead to a control action.

Fortunately, the temporal correlation between the subsequent samples of the PEE is very high and step changes in the PEE do not occur very frequently. As a result, there is more time available than the time between two subsequent position updates to smoothen out this effect. There are several ways to benefit from this. The most simple one is to put a fixed limit on the maximum correction gradient. The question then becomes how to select the magnitude of this limit. For the current implementation of the datashaper, the limit is dynamic. It is based on a prediction of the maximum errors that can occur in the dead-reckoning part. Based on estimates of the maximum velocity error, the maximum track error and the maximum flightpath angle error, a 3-D volume is computed in which the new position update from the positioning system is expected to occur. If the new update lies outside this volume, it is assumed that this is due to a change in the PEE. In this case, only the error component represented by $\bar{e}$ (Fig. 10) will be distributed over the extrapolations until the next position update. The vector $\bar{e}$ used for smoothing, is determined by the intersection of a line between the predicted position and the new position estimate with the 3-D volume.

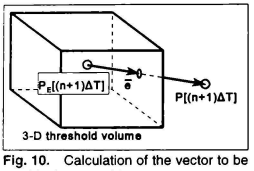


Fig. 10. Calculation of the vector to be used in the smoothing process.

The modified datashaper

The structure of the modified datashaper is presented in Fig. 11.

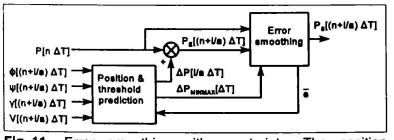


Fig. 11. Error smoothing with constraints. The position prediction calculates a maximum threshold ΔP_{MAX} for the expected changes in the position during an interval ΔT. If ΔP exceeds ΔP_{MINMAX}, the error smoother uses the values of $\Delta P_{MINMAX}/a$ during each update.

There are two additional datapaths as compared to the datashaper presented in Fig. 7. The threshold prediction provides the smoother with a 3-D volume in which the expected position update is expected to occur. If the new update occurs outside the volume, it is expected that this is caused by a change in PEE. The ratio of the change in PEE and the size of the constraints of the 3-D volume determine the time that is taken to smooth out the step change. However, in certain situations, an error may accumulate undetected. Therefore, a feedback from the error smoother to the threshold predictor is implemented. If a subsequent (predefined) number of position updates occurs outside this volume, the feedback from the smoother to the threshold prediction increases the uncertainty margins used in the prediction.

Other options to reduce the position error gain

As mentioned earlier, the position error gain is determined by the tunnel size. However, this is only the case when the tunnel is fixed in space. For example, a tunnel that is continuously regenerated and always starts at the actual location of the aircraft while pointing in the desired direction of flight does not provide any feedback on the current position error at all. By using the concept of a tunnel that is not fixed in space, it might also be possible to use a size that provides sufficient temporal range information while at the same time the position error gain can be reduced by a specified ratio. This option is not further explored in this paper.

EVALUATION

Goals

The goals of the experiment were:

1. To obtain quantitative data on the likelihood of detection of distortions in the optic flow pattern by the pilot.
2. To obtain quantitative data on the influence of the position sensor error on pilot control activity and tracking performance
3. To determine whether the likelihood of detection of anomalies in the optic flow pattern can be reduced by means of the datashaper proposed in this paper.

Experimental design

To satisfy the goals of the experiment, a within subject repeated measures design was used. The following three conditions were used:

0. True position, 30 Hz.
1. Position from datashaper, 30 Hz.
2. GPS position, 5 Hz.

Condition 0 served as the reference condition to establish a baseline for tracking performance and control activity. Conditions 0 and 1 were presented in a balanced random order to average out remaining learning effects. Subjects were not informed on the condition. To prevent subjects from becoming accustomed to a particular trajectory, four different trajectories were used. The third condition served as a reference to determine the likelihood of detection of anomalies in the optic flow pattern in the absence of smoothing.

Setup

The experiment was performed in a part-task simulator setup. The aircraft being simulated was a twin engine jet with a mass of 20,000 kg. The control system was an attitude rate command/attitude hold system. Control devices consisted of a side stick and two throttle levers. To present the pilot with a PFD (Fig. 12), a navigation display (ND) and an engine display, three flat panel displays were used.

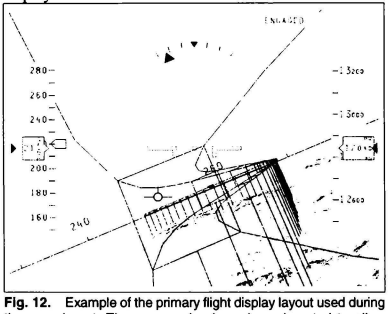


Fig. 12. Example of the primary flight display layout used during the experiment. The gray scales have been inverted to allow better reproduction on paper.

To provide the position data as would normally be obtained from a GPS receiver, a GPS simulator was integrated with the setup. SA was simulated according to the model discussed by Van Graas and Braasch[5]. To obtain a sufficient number of measurements on the effect of changes in the PEE, the frequency of occurrence of PEE step changes was increased.

Display configuration

The PFD was the basic DELPHINS tunnel-in-the-Sky display format[13] and included an airspeed tape, an altitude tape, a heading tape, a roll angle indicator, an aircraft attitude symbol, and a flight path vector (Fig. 12). As discussed earlier, the size of the tunnel determines the position error gain and thus will influence the amount of control activity that results from position sensor errors. As mentioned in the requirements analysis, no validated pilot model is available that can be used to select a maximum position error gain based on control motion noise restrictions. Therefore, an educated guess will be required with respect to the initial values of the tunnel size. Based on the results from previous experiments[12,13] and a small pilot study into the effect of position sensor errors, a tunnel size of 90 m was selected.

Datashaper configuration

The datashaper used a first order predictor. The margins for the velocity error were set at 20 kts, for the track error at 10 degrees, and for the flightpath angle error at 5 degrees.

Subjects

Seven subjects participated in the experiment. Two of the subjects possessed a pilot's license and were familiar with the DELPHINS tunnel-in-the-sky display. One other subject had extensive experience in flight simulators and with the DELPHINS tunnel-in-the-sky display. The other four subjects had never flown a tunnel-in-the-sky display before.

Training

To familiarize the subjects with the displays, the controls, and the dynamics of the system, training flights were performed. In order to minimize the learning effect during the experiment trials, tracking performance and subject opinion were used to determine whether the subject was sufficiently trained.

Task

The task of the subjects was to fly the aircraft in the presence of a wind of approximately 15 kts at a commanded velocity of 220 kts along a trajectory consisting of straight- and curved segments. The trajectory also included climb- and descent segments. It was presented on the PFD and the ND. Subjects were instructed to fly as accurate as possible using the PFD, and to push the trigger on the sidestick when they detected an unexpected motion relative to the flightpath.

Rating parameters

As mentioned earlier, a change in PEE should not result in a control action. To determine the amount of control activity, stick deflection was recorded. To compare tracking performance between conditions (0) and (1), the position steering error was also recorded. To quantify the ability of the datashaper to smoothen out the effects of sudden changes in the position error, the detection ratio of visual anomalies in the 5 Hz condition (2) was compared with the detection ratio of the datashaper condition (1).

Results

The detection ratio of visual anomalies caused by changes in the PEE is reduced from 84% to 23%. To gain more insight into the relation between the magnitude of the PEE change and the likelihood of detection, Fig. 13 shows the detection ratio for conditions (1) and (2) as a function of the size of magnitude of the change in PEE.

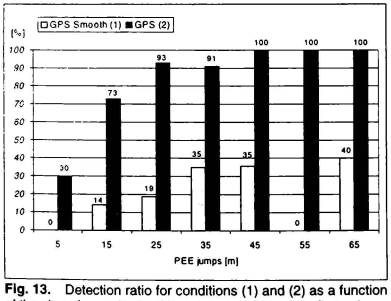


Fig. 13. Detection ratio for conditions (1) and (2) as a function of the size of magnitude of the change in PEE. The first column represents changes in PEE between 0 and 10 m, the second between 10 and 20 m, etc.

As can be seen from Fig. 13, approximately 93% of changes in position error between 20 and 30 [m] are detected in the absence of smoothing. The datashaper reduces this to less than 20%.

As mentioned in the subsection about position error gain ans SA, the position error gain should be selected low enough that the changes in PEE caused by SA do not lead to pilot control actions. If this is the case with a tunnel of 90 m, there should be no significant difference in both tracking performance and control activity between conditions (0) and (1). Fig. 14 shows the lateral tracking performance for conditions (0) and (1), and Fig. 15 shows the aileron control activity for conditions (0) and (1).

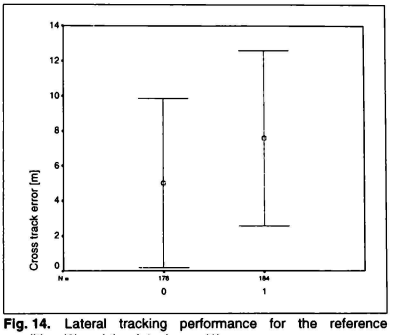


Fig. 14. Lateral tracking performance for the reference condition (0) and the datashaper (1).

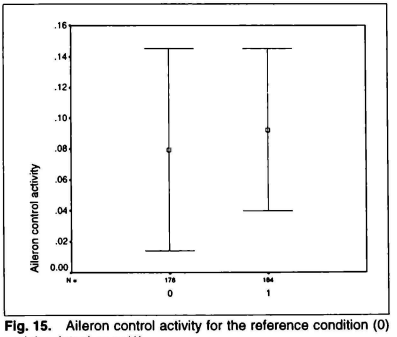


Fig. 15. Aileron control activity for the reference condition (0) and the datashaper (1).

A statistical analysis showed that there are no significant differences at a 5% significance level.
Fig. 16 shows the vertical tracking performance for conditions (0) and (1), and Fig. 17 shows the elevator control activity for conditions (0) and (1).

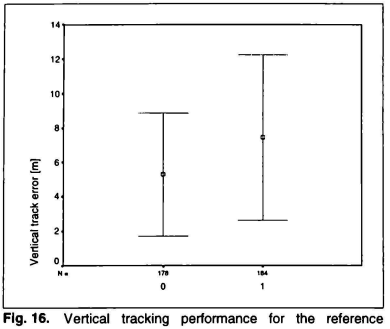


Fig. 16. Vertical tracking performance for the reference condition (0) and the datashaper (1).

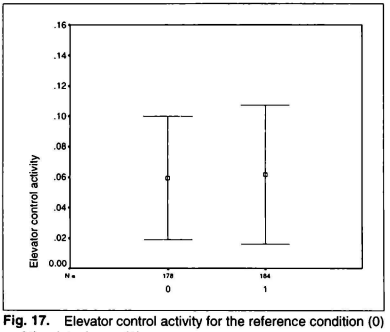


Fig. 17. Elevator control activity for the reference condition (0) and the datashaper (1).

A statistical analysis revealed a significant difference in vertical tracking performance, but not in elevator control activity.

Discussion

The results indicate that the datashaper yields a significant reduction in the detection of PEE induced distortions of the optic flow pattern. The results presented in Fig. 13 were obtained using a tunnel with a width and a height of 90 m, but can be translated to other position error gains. When doubling the size of the tunnel, the position error gain is halved, and the results presented in Fig. 13 can be translated to this situation by doubling the numbers on the x-axis.
With respect to lateral tracking performance and pilot control activity, the fact that no significant differences were found between the conditions leads to the conclusion that no significant increase in control motion noise is introduced with the current datashaper and a tunnel size of 90 m. Only in the vertical direction a significant difference in tracking performance was found. A more detailed analysis of the data is needed to determine whether this was caused by a too high position error gain, a too large value of the vertical threshold, or another factor.

CONCLUSIONS

When using unaugmented GPS position data with an update rate of 5 Hz (MOPS for GPS/WAAS[10]) to generate an egocentric perspective flightpath display, additional measures must be taken in order to satisfy perceptual requirements as specified in AC 25-11[1].
Initial results from the experiment indicate that the concept of extrapolation and smoothing as described in this paper can be used to satisfy the perceptual requirements that must be met when using perspective flightpath displays.

ACKNOWLEDGEMENTS

This research is sponsored by the Dutch Technology Foundation STW in the context of the Delft Program for Hybridized Instrumentation and Navigation Systems (DELPHINS) phase II. The authors would also like to thank Richard Rademaker for his improvements to the DELPHINS part-task simulator and Bas Ober for his help with the GPS simulator.

REFERENCES

1. AC 25-11 (1987). Transport Category Airplane Electronic Display Systems, U.S. Department of Transportation, Federal Aviation Administration.
2. Battiste, V., Downs, M., Sullivan, B. and Soukup, P. (1996). 'Utilization of a Ground Taxi Map to Support Low Visibility Surface Operations', *Proceedings of the AIAA Flight Simulation Technologies Conference*, pp. 393-403.
3. Ethell, J.L. (1994). NASA's blueprint for a general aviation renaissance. *Aerospace America*, Oct. 1994, pp.38-43.
4. Funk, J.D., Beck, C.P., and Johns, J.B. (1993). 'Primary Display Latency Criteria Based on Flying Qualities and Performance Data', *Proceedings of 'Pilot Vertical Flight Aircraft: A Conference on Flying Qualities and Human Factors'*, pp. 4-77 to 4-90.
5. Graas. F. van, and Braasch, M.S. (1996). Selective Availability. In: Global Positioning System: Theory and Applications, Volume 1. Eds. Parkinson, B.J. and Spilker, J.J.
6. Grunwald, A.J. (1981). 'Experimental Evaluation of a Perspective Tunnel Display for Three-Dimensional Helicopter Approaches', *Journal of Guidance and Control*, Vol. 4, No. 6, pp. 623-631.
7. Johns, L. (1988). 'A study of the effects delay times in a dome-to-dome simulation link', *Proceedings of the AIAA Flight Simulation Technologies Conference*, pp. 122-127.
8. Katz, A. and Graham, K. (1994). 'Extrapolation of Airplane States' , *Proceedings of the AIAA Flight Simulation Technologies Conference*, pp. 228-236.
9. Lin, K., Wang, M., Wang, J., Schab, D.E. (1995). 'The smoothing of dead reckoning image in distributed interactive simulation', *Proceeedings of the AIAA Flight Simulation Technologies Conference*, pp. 83-87.
10. RTCA DO-229 (1996). Minimum Operational Performance Standards for Global Positioning System/Wide Area Augmentation System Airborne Equipment.
11. Sachs and Möller (1995). 'Synthetic Vision Flight Tests for Precision Approach and Landing', *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pp. 1459-1466.
12. Theunissen (1993). 'A Primary Flight Display for Four-Dimensional Guidance and Navigation - Influence of Tunnel Size and Level of Additional Information on Pilot Performance and Control Behaviour', *Proceedings of the AIAA Flight Simulation Technologies Conference*, pp. 140-146.
13. Theunissen (1997). Integrated Design of a Man-Machine Interface for 4-D Navigation, ISBN 90-407-1406-1, Delft University Press, Delft, The Netherlands.
14. Wickens, C.D., Haskell, I., and Harte, K. (1989). 'Ergonomic Design for Perspective Flightpath Displays', *IEEE Control Systems Magazine*, pp. 3-8.

LANGUAGE-BASED MODELING AND SIMULATION IN VERILOG-A: A CASE STUDY OF GIMBAL SYSTEMS

Reza Ahmadi
Raytheon Systems Co.
El Segundo, CA

Russell D. Smith
Raytheon Systems Co.
Tewksbury, MA

Larry Rubin and Amir M. Zarkesh
Transcendent Design Technology, Inc.
Camarillo, CA

Abstract

In this paper we introduce a powerful new standard behavioral/structural language, Verilog-A, for system simulation. We present the open-loop analysis of multi-gimbal systems and discuss the advantages of adding behavioral modeling capabilities to standard hierarchical structural modeling. Explicit *n*-gimbal dynamic equations are developed in terms of momentum states that might be directly implemented in any of several commercially available systems simulation tools. By way of contrast, as a specific case study, we write the behavioral models for a three-gimbal system in TransVerSE™ extended Verilog-A language by instantiating "n" gimbals and defining how they are connected together in an appropriately compact form. We also summarize the main features of the next generation language-based simulation methodologies that are currently emerging due to the availability of standard hardware description languages.

I. Introduction

In addition to its vital role in control systems design and analysis, system simulation has found an increasingly important role in shortening the design cycles and lowering the cost of complex aerospace systems. Today, there are many commercially available simulation tools, like Matlab™ and EASY5™, which provide a wide range of computational capabilities. The methodology of capturing a system in these simulators is based on a hierarchical connection of "components" that are either provided in their libraries directly, or can be added through the facility of user-defined algorithms or macro code blocks. This methodology provides an efficient way to reuse the component models and minimize duplication of effort. New components can be constructed by connecting the primitive components. The main bottleneck in using these popular simulation tools is the large investment time and effort they demand for the modeling of new systems. Behavioral modeling, by contrast, is a powerful means of capturing essential features without the need of accessing all structural details. This is especially important in the early stages of the design where the structural details are not yet determined, and only the expected behavior is known.

However, the complexity of describing behavioral models in terms of structural primitives or behavioral descriptions increases rapidly. Currently the gap between the behavioral descriptions understood by engineers in terms of equations and algorithms about hardware on the one hand, and the forms and constructs acceptable to simulation tools like EASY5™ and Matlab™ on the other, is very wide. A substantial amount of expertise and time is needed to convert the hardware description understood by the engineers to appropriate signal-flow diagrams which are suitable to be simulated by the existing simulators.

A natural medium to express behavior is in a language, since the engineers think about the intended behaviors of their systems in terms of equations and algorithms in a language. However, instead of a general mathematical language we need a suitable hardware description language. In the past few years, an effort has been underway to extend the scope of the popular digital hardware description languages, VHDL and Verilog, to cover analog and mixed signal domains. The result is VHDL-AMS, which has been standardized recently by IEEE, and Verilog-AMS which is being finalized by Open Verilog International (OVI); this latter will be sent to IEEE sometime during the summer of 1999. Verilog-A provides the means for behavioral and structural modeling of systems across the different energy domains independently of any specific simulation tool. Essentially, any system that can be described by a set of coupled differential-algebraic equations can be modeled in Verilog-A.

One of the crucial features of Verilog-A is the capability to combine *signal-flow* modeling with *conservative* modeling. This capability helps the modeling effort in a significant manner by giving the modeler an ability to think about the physical description rather than translating it to bare mathematical relations. Moreover, implicit equations are supported as well as explicit equations. These features allow a concise way to write, for example, behavioral models for gimbal systems. This

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

methodology provides the means to simulate a system from early design stages, when only the expected behavior of the subsystems are known, to the later stages when the structural details of the subsystems have been designed.

In this paper, we consider the simulation of the dynamics of a multi-gimbal system. Such a task has applications in system simulations where the objective is to keep a platform stable with respect to inertial space or to command the orientation: for example, an Inertial Navigation System (INS) in a stabilization mode or a camera on a moving vehicle. The mechanization of an INS system often involves the rotational stabilization of a gimbaled platform where the gyros are mounted. To keep the platform at a fixed orientation with respect to inertial space or local level geographic coordinates, the output of the gyros (inertial angular velocity) are used to command the torque motors which turn the gimbals in such a way as to null the gyro outputs. The same model can be applied to a camera mounted in a moving vehicle.

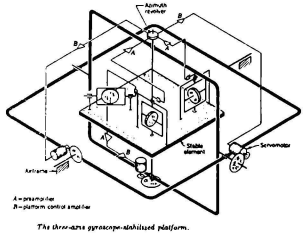


Figure 1: Three-axis gyro-controlled platform

For the sake of brevity, however, our focus in this introductory paper will be restricted to the open-loop case; i.e., gimbal torque motors won't make use of rate or angular feedback.

The organization of the rest of this paper is as follows. First we present the explicit gimbal dynamic equations in terms of angular momentum states as expressed in body-fixed frames. This formulation allows us to systematically solve for gimbal axis inertial angular velocities as an explicit function of state-derived and input quantities; the algorithms were implemented in both Matlab™ and EASY5™ for reference. Next, we formulate the *n*-gimbal problem implicitly using the TransVerSE™ extensions of the Verilog-A language, and we discuss the meaning and advantages of conservative and object-oriented simulation. Finally, we present the simulation results for an open-loop 3-gimbal system, and discuss further some of the computational distinctions.

II. Explicit Multi-Gimbal Dynamics

For a general set of rigid gimbals, the (absolute) angular momentum relation provides the standard rotational equations of motion, with all vectors expressed in body-fixed frames[1]:

$$\dot{h}_i + (\omega_i \times h_i) = \sum \tau_i + (m_i v_i \times r_i^c \times \omega_i) \qquad (1)$$

where
$$h_i = \bar{I}_i \omega_i + (m_i r_i^c \times v_i) \qquad (2)$$

$$\bar{I}_i = I_i^c - m_i [r_i^c \times][r_i^c \times] \qquad (3)$$

$$v_i = R_0^i v_0 \qquad (4)$$

We shall retain the momentum variables in the equations of motion, as indicated by Eq. (1), and retrieve the desired velocity variables from the algebraic relation of Eq. (2); this eases the mathematical notation that follows, and eliminates the need for numerically inverting. Note that all time-derivatives indicate rates of change with respect to the body-fixed gimbal frames, and translational motion is a kinematic driving input. Given the classical nature of the problem, the notation is largely implied; but in Eq. (3), $[r_i^c \times]$ is a skew-symmetric cross-product matrix formed from the components of $r_i^c = [x_i^c \; y_i^c \; z_i^c]^T$, which locate the gimbal CG with respect to the gimbal frame reference origins; I_i^c are the associated centroidal inertia matrices. The only simplifying assumption to be made in the application of Eqs. (1)-(4), beyond the single inherent rigidity assumption, is that gimbal rotational axes are mutually intersecting – the common non-centroidal point of reference for all of the body-fixed gimbal vector quantities. If/when this might not be the case, one may introduce the additional momentum terms by augmenting the velocity relation of Eq. (4)[2,3].

Since the applied torques may be summarized by a term representing an exterior gimbal and the reaction torque from an interior gimbal, one has:

$$\dot{h}_i = \tau_i - R_{i+1}^i \tau_{i+1} - (\omega_i \times h_i) + (m_i v_i \times r_i^c \times \omega_i) \qquad (5)$$

The numbering system has been chosen such that gimbal index numbers are increasing as one proceeds from exterior to interior gimbals. A scheme of orthogonal frame orientations will now be adopted for consistency among all gimbals: torque motor rotational axes (i.e., the degree-of-freedom axes) will be taken as body-fixed x-axes; at zero gimbal angle conditions, these are oriented in

the directions of the body-fixed z-axis of exterior gimbals, and in the body-fixed y-axis of interior gimbals. Thus,

$$R_i^{i+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_{i+1} & -\sin\theta_{i+1} \\ 0 & \sin\theta_{i+1} & \cos\theta_{i+1} \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R_i^{i+1} = \begin{bmatrix} 0 & 0 & 1 \\ \cos\theta_{i+1} & -\sin\theta_{i+1} & 0 \\ \sin\theta_{i+1} & \cos\theta_{i+1} & 0 \end{bmatrix} \quad (6)$$

with gimbal angle relation,

$$\dot{\theta}_{i+1} = \{R_i^{i+1}\omega_i - \omega_{i+1}\}_x = \omega_{iz} - \omega_{(i+1)x} \quad (7)$$

Therefore, in Eq. (5), the control torques are:

$$\tau_{ix} = T_i^M - \tau_i^F = T_i^{Net} \quad (8)$$

and the constraint torques, identically satisfying Eq. (5), are computed from:

$$\tau_{iy} = \{\dot{h}_i + (\omega_i \times h_i) - (m_i v_i \times r_i^c \times \omega_i) + R_{i+1}^i \tau_{i+1}\}_y \quad (9)$$

$$\tau_{iz} = \{\dot{h}_i + (\omega_i \times h_i) - (m_i v_i \times r_i^c \times \omega_i) + R_{i+1}^i \tau_{i+1}\}_z \quad (10)$$

where $\{\cdot\}_{x,y,z}$ indicates the x-, y-, or z-component of the vector argument, and where any particular gimbal frame's x-axis experiences the net of the control motor torque less any rotational axis friction. In view of the scheme of Eq. (6), it is noted that the component of reaction torque along any gimbal rotation axis involves only sub-matrix multiplication: i.e., in Eq. (5),

$$\begin{aligned} \{R_{i+1}^i \tau_{i+1}\}_x &= [0 \quad \cos\theta_{i+1} \quad -\sin\theta_{i+1}]\tau_{i+1} \\ &= [R_{i+1}^i{}_{(1,2)} \quad R_{i+1}^i{}_{(1,3)}][\tau_{(i+1)y} \quad \tau_{(i+1)z}]^T \quad (11) \\ &= \overline{R}_{i+1}^i [\tau_{(i+1)y} \quad \tau_{(i+1)z}]^T \end{aligned}$$

It is also to be noted that the adoption of the foregoing scheme of gimbal frame relations introduces an orthogonality assumption, not inherent in the dynamics of Eq. (1), and has only been made for the convenience of the mathematical presentation. Mechanical misalignments might well be addressed with unique, and perhaps non-orthogonal transformation matrices between gimbals.

3-Gimbal System

Applying Eq. (5) sequentially for $i = 1 \rightarrow 3$, and extracting the degrees-of-freedom from the first rows yields:

$$\dot{h}_1 = T_1^{Net} - \{R_2^1\tau_2 + (\omega_1 \times h_1) - (m_1 v_1 \times r_1^c \times \omega_1)\}_x \quad (12)$$

$$\dot{h}_2 = T_2^{Net} - \{R_3^2\tau_3 + (\omega_2 \times h_2) - (m_2 v_2 \times r_2^c \times \omega_2)\}_x \quad (13)$$

$$\dot{h}_3 = T_3^{Net} - \{\qquad (\omega_3 \times h_3) - (m_3 v_3 \times r_3^c \times \omega_3)\}_x \quad (14)$$

which, upon substitution of Eqs. (8)-(10), becomes:

$$\tilde{A}\dot{\tilde{h}} = b^* \quad (15)$$

where

$$\begin{aligned} \tilde{h} &= [h_1 \; h_2 \; h_3]^T \\ &= [h_{1x} \; h_{1y} \; h_{1z} \mid h_{2x} \; h_{2y} \; h_{2z} \mid h_{3x} \; h_{3y} \; h_{3z}]^T \end{aligned}$$

and

$$\tilde{A} = \left[\begin{array}{ccc|c|c} 1 & 0 & 0 & [0 & \overline{R}_1^1] & [0 & (\overline{R_2^1 R_3^2})] \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & [0 & \overline{R}_3^2] \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right]$$

$$b^* = \begin{bmatrix} T_1^{Net} - (\omega_1 \times h_1)_x + (m_1 v_1 \times r_1^c \times \omega_1)_x - \overline{R}_2^1[\tau_{2y}^* \quad \tau_{2z}^*]^T \\ T_2^{Net} - (\omega_2 \times h_2)_x + (m_2 v_2 \times r_2^c \times \omega_2)_x - \overline{R}_3^2[\tau_{3y}^* \quad \tau_{3z}^*]^T \\ T_3^{Net} - (\omega_3 \times h_3)_x + (m_3 v_3 \times r_3^c \times \omega_3)_x \end{bmatrix}$$

The cumulative remainder reaction "torques" τ_i^* are:

$$\begin{bmatrix} \tau_{ix}^* \\ \tau_{iy}^* \\ \tau_{iz}^* \end{bmatrix} = \begin{bmatrix} T_i^{Net} \\ \{(\omega_i \times h_i) - (m_i v_i \times r_i^c \times \omega_i) + R_{i+1}^i \tau_{i+1}^*\}_y \\ \{(\omega_i \times h_i) - (m_i v_i \times r_i^c \times \omega_i) + R_{i+1}^i \tau_{i+1}^*\}_z \end{bmatrix} \quad (16)$$

It now remains to express the dependent momentum variables in terms of the independent variables, and to explicitly substitute these into Eq. (15).

In view of the form of Eq. (6) and (7), kinematics propagate as:

$$\begin{aligned} \omega_i &= R_{i-1}^i \omega_{i-1} - [\dot{\theta}_i \quad 0 \quad 0]^T \\ &= R_{i-1}^i \omega_{i-1} - [(\omega_{(i-1)z} - \omega_{ix}) \quad 0 \quad 0]^T \end{aligned}$$

The angular velocities are now partitioned among independent and dependent variables:

$$[\varpi_x \; \varpi_{yz}] = [\omega_{1x} \; \omega_{2x} \; \omega_{3x} \mid \omega_{1y} \; \omega_{1z} \; \omega_{2y} \; \omega_{2z} \; \omega_{3y} \; \omega_{3z}]$$

Thus, the dependent velocities and accelerations may be propagated, recursively in the y-components, as follows:

$$\begin{bmatrix} \omega_{1y} \\ \omega_{1z} \\ \omega_{2y} \\ \omega_{2z} \\ \omega_{3y} \\ \omega_{3z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ R_1^2(2,1) & 0 & 0 \\ R_1^2(3,1) & 0 & 0 \\ (R_2^3 R_1^2)(2,1) & R_2^3(2,1) & 0 \\ (R_2^3 R_1^2)(3,1) & R_2^3(3,1) & 0 \end{bmatrix} \begin{bmatrix} \omega_{1x} \\ \omega_{2x} \\ \omega_{3x} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} R_1^1(2,1) & R_1^1(2,2) \\ R_1^1(3,1) & R^1(3,2) \end{bmatrix} \begin{bmatrix} \omega_{1x} \\ \omega_{1y} \end{bmatrix} \\ R_1^2(2,2)\omega_{1y} \\ R_1^2(3,2)\omega_{1y} \\ R_2^3(2,2)\omega_{2y} \\ R_2^3(3,2)\omega_{2y} \end{bmatrix}$$

or,

$$\varpi_{yz} = C\varpi_x + \chi \quad (17)$$

and

$$\begin{bmatrix}\dot{\omega}_{1y}\\ \dot{\omega}_{1z}\\ \dot{\omega}_{2y}\\ \dot{\omega}_{2z}\\ \dot{\omega}_{3y}\\ \dot{\omega}_{3z}\end{bmatrix} = \begin{bmatrix} 0 & 0 & 0\\ 0 & 0 & 0\\ R_1^2{}_{(2,1)} & 0 & 0\\ R_1^2{}_{(3,1)} & 0 & 0\\ (R_2^1R_1^2)_{(2,1)} & R_2^1{}_{(2,1)} & 0\\ (R_2^1R_1^2)_{(3,1)} & R_2^1{}_{(3,1)} & 0\end{bmatrix}\begin{bmatrix}\dot{\omega}_{1x}\\ \dot{\omega}_{2x}\\ \dot{\omega}_{3x}\end{bmatrix} + \begin{bmatrix}\dot{\omega}'_{1y}\\ \dot{\omega}'_{1z}\\ \dot{\omega}'_{2y}\\ \dot{\omega}'_{2z}\\ \dot{\omega}'_{3y}\\ \dot{\omega}'_{3z}\end{bmatrix}$$

or,

$$\dot{\varpi}_{yz} = C\dot{\varpi}_x + \dot{\omega}' \qquad (18)$$

where $\omega_{0x,y}$ refers to base motion input, and where the "primed" component of the dependent accelerations is:

$$\begin{bmatrix}\dot{\omega}'_{1y}\\ \dot{\omega}'_{1z}\\ \dot{\omega}'_{2y}\\ \dot{\omega}'_{2z}\\ \dot{\omega}'_{3y}\\ \dot{\omega}'_{3z}\end{bmatrix} = \begin{bmatrix} R_0^1{}_{(2,1)} & R_0^1{}_{(2,2)}\\ R_0^1{}_{(3,1)} & R_0^1{}_{(3,2)}\\ R_1^2{}_{(2,2)} & \\ R_1^2{}_{(3,2)} & \\ R_2^3{}_{(2,2)} & \\ R_2^3{}_{(3,2)} & \end{bmatrix}\begin{bmatrix}\dot{\omega}_{0x}\\ \dot{\omega}_{0y}\end{bmatrix}\dot{\theta}_1 + \frac{\partial}{\partial\theta_1}\begin{bmatrix} R_0^1{}_{(2,1)} & R_0^1{}_{(2,2)}\\ R_0^1{}_{(3,1)} & R_0^1{}_{(3,2)}\end{bmatrix}\begin{bmatrix}\omega_{0x}\\ \omega_{0y}\end{bmatrix}\dot{\theta}_1 + \begin{bmatrix} R_1^2{}_{(2,2)}\\ R_1^2{}_{(3,2)}\end{bmatrix}\dot{\omega}'_{1y} + \frac{\partial}{\partial\theta_2}\begin{bmatrix} R_1^2{}_{(2,2)}\\ R_1^2{}_{(3,2)}\end{bmatrix}\begin{bmatrix}\omega_{1x}\\ \omega_{1y}\end{bmatrix}\dot{\theta}_2 + \begin{bmatrix} R_2^3{}_{(2,2)}\\ R_2^3{}_{(3,2)}\end{bmatrix}\dot{\omega}'_{2y} + \frac{\partial}{\partial\theta_3}\begin{bmatrix} R_2^3{}_{(2,1)} & R_2^3{}_{(2,2)}\\ R_2^3{}_{(3,1)} & R_2^3{}_{(3,2)}\end{bmatrix}\begin{bmatrix}\omega_{2x}\\ \omega_{2y}\end{bmatrix}\dot{\theta}_3$$

Now, partitioning the x-axis moments and products of inertia:

$$[\bar{I}_x\ \bar{I}_{yz}] = \left[\begin{array}{ccc|ccccc} \bar{I}^1_{xx} & 0 & 0 & \bar{I}^1_{xy} & \bar{I}^1_{xz} & 0 & 0 & 0 & 0\\ 0 & \bar{I}^2_{xx} & 0 & 0 & 0 & \bar{I}^2_{xy} & \bar{I}^2_{xz} & 0 & 0\\ 0 & 0 & \bar{I}^3_{xx} & 0 & 0 & 0 & 0 & \bar{I}^3_{xy} & \bar{I}^3_{xz}\end{array}\right]$$

Then extracting the independent rows of Eq. (2), upon substitution of Eq. (17), yields:

$$[\bar{I}_x + \bar{I}_{yz}C]\dot{\varpi}_x = \begin{bmatrix}\dot{h}_{1x}\\ \dot{h}_{2x}\\ \dot{h}_{3x}\end{bmatrix} - \begin{bmatrix}\{m_1r_1^c \times v_1\}_x\\ \{m_2r_2^c \times v_2\}_x\\ \{m_3r_3^c \times v_3\}_x\end{bmatrix} - \bar{I}_{yz}\chi \qquad (19)$$

Also,

$$[\bar{I}_x + \bar{I}_{yz}C]\dot{\varpi}_x = \begin{bmatrix}\dot{h}_{1x}\\ \dot{h}_{2x}\\ \dot{h}_{3x}\end{bmatrix} - \begin{bmatrix}\{m_1r_1^c \times \dot{v}_1\}_x\\ \{m_2r_2^c \times \dot{v}_2\}_x\\ \{m_3r_3^c \times \dot{v}_3\}_x\end{bmatrix} - \bar{I}_{yz}\dot{\omega}' \qquad (20)$$

where the coefficient matrix on the left hand side of Eq. (20) is lower triangular, and may be solved with back-substitution, or by analytically inverting. Thus, Eqs. (17)-(20) explicitly provide all angular velocities and accelerations in terms of the independent angular momentum states and state derivatives, and the propagated kinematic forcing functions stemming from base motion input. Note that since χ requires a sequential computation, so do the elements of ϖ_x and ϖ_{yz}. Furthermore, since the translational motion derives from kinematic driving inputs, v_0 and a_0, Eq. (20) implies the further substitution:

$$\dot{v}_i = R_0^i a_0 - (\omega_i \times R_0^i v_0) \qquad (21)$$

The dependent momentum variables are now computed from:

$$h_{yz} = \bar{I}_x\varpi_x + \bar{I}_{yz}\varpi_{yz} \qquad (22)$$

$$\dot{h}_{yz} = \bar{I}_x\dot{\varpi}_x + \bar{I}_{yz}\dot{\varpi}_{yz} \qquad (23)$$

where

$$h_{yz} = \begin{bmatrix}h_{1y}\\ h_{1z}\\ h_{2y}\\ h_{2z}\\ h_{3y}\\ h_{3z}\end{bmatrix} \qquad \bar{I}_x = \left[\begin{array}{c|c|c} \begin{bmatrix}\bar{I}^1_{yx}\\ \bar{I}^1_{zx}\end{bmatrix} & 0 & 0\\ \hline 0 & \begin{bmatrix}\bar{I}^2_{yx}\\ \bar{I}^2_{zx}\end{bmatrix} & 0\\ \hline 0 & 0 & \begin{bmatrix}\bar{I}^3_{yx}\\ \bar{I}^3_{zx}\end{bmatrix}\end{array}\right]$$

$$\bar{I}_{yz} = \left[\begin{array}{c|c|c} \begin{bmatrix}\bar{I}^1_{yy} & \bar{I}^1_{yz}\\ \bar{I}^1_{zy} & \bar{I}^1_{zz}\end{bmatrix} & 0 & 0\\ \hline 0 & \begin{bmatrix}\bar{I}^2_{yy} & \bar{I}^2_{yz}\\ \bar{I}^2_{zy} & \bar{I}^2_{zz}\end{bmatrix} & 0\\ \hline 0 & 0 & \begin{bmatrix}\bar{I}^3_{yy} & \bar{I}^3_{yz}\\ \bar{I}^3_{zy} & \bar{I}^3_{zz}\end{bmatrix}\end{array}\right]$$

Re-partitioning Eq. (15) according to:

$$\tilde{A}\ddot{h} = [A_x \mid A_{yz}][\dot{h}_{1x}\ \dot{h}_{2x}\ \dot{h}_{3x} \mid \dot{h}_{1y}\ \dot{h}_{1z}\ \dot{h}_{2y}\ \dot{h}_{2z}\ \dot{h}_{3y}\ \dot{h}_{3z}]^T$$

and substituting from Eqs. (18), (20), and (23) yields the following explicit system:

$$\left[A_x + A'_{yz}\begin{bmatrix}\dot{h}_{1x}\\ \dot{h}_{2x}\\ \dot{h}_{3x}\end{bmatrix}\right] = b^* - \dot{h}' \qquad (24)$$

where

$$\dot{h}' = A_{yz}\tilde{I}_{yz}\dot{\varpi}_{yz} - A'_{yz}\left\{\begin{bmatrix}\{m_1r_1^c \times \dot{v}_1\}_x\\ \{m_2r_2^c \times \dot{v}_2\}_x\\ \{m_3r_3^c \times \dot{v}_3\}_x\end{bmatrix} + \bar{I}_{yz}\dot{\omega}'\right\} \qquad (25)$$

$$A'_{yz} = A_{yz}\tilde{I}_x[\bar{I}_x + \bar{I}_{yz}C]^{-1} \qquad (26)$$

and the coefficient matrix on the left hand side of Eq. (24) is now upper triangular, where A_x is merely the 3rd-order identity matrix. The presentation of the dynamics here, in terms of momentum variables, is not substantially different than typical forms of the recursive schemes; one may "outwardly" propagate the kinematics of Eqs. (17)-(20), and "backwardly" compute the dynamics of Eq. (24). Or the inverse indicated in Eq. (26), and the other implied in Eq. (24), may be computed analytically, for triangular matrices, and the dynamics may be solved simultaneously.

III. Implicit Multi-Gimbal Dynamics and Conservative Simulation

There are at least two crucial advantages that explicit modeling has when compared to implicit modeling: first, standard simulation tools may be used to implement the models; and second, they can be expected to execute relatively quickly. However, a significant drawback is the labor involved in mathematically formulating the explicit solution from the original problem statement. Moreover, the debugging of the explicit model is relatively more difficult since the analyst has rather substantially departed from the physical description. One goal of hardware description languages like Verilog-A[4] or VHDL-A is to provide an environment where the models have the closest content to the way that engineers understand the physical system.

The physical description of many systems is sometimes more natural if it is done in a mixed *conservative* and *non-conservative*, or signal-flow, paradigm. Signal-flow modeling is a classic methodology used for the simulation of control and many other systems. In this methodology, quantities are free to take any values permitted by the *assignment-statement* equations provided by the user. In many applications, there are physical conservation laws that we intend should hold; for example, the conservation of currents at an electrical node. In a mechanical system, the net of the forces/torques acting on a body should equal the time rate of change of its linear/angular momentum. If one wants to simulate such systems in a *signal-flow* methodology, the user must mathematically formulate all of these conservation laws. By contrast, in a conservative simulation paradigm, each access "node" possesses two inherent, power-conjugate quantities – a flow variable and a potential variable. In the electrical energy domain, flow and potential are commonly taken as node currents and node voltages; in the mechanical domain, the flow and potential variables can be considered as force/torque and linear/angular velocity[5]. A conservative simulator enforces the conservation law of the zero sum of the flow to each node automatically. In this paradigm, the user will instead concentrate his attention on the constitutive laws, or behaviors of the system.

The lack of matrix calculations in VHDL-A is one of the reasons why it might be less suitable for modeling systems with complex dynamics. The multi-dimensional extensions of TransVerSE make the compact representation of vector equations possible. Algebraic expressions involving bracketed variables are always understood as matrix equations. As can be seen from the gimbal model given in the Appendix, the written equations are intended to closely resemble the dynamic equations that first come to mind when modeling the gimbals. Moreover, the object-oriented nature of Verilog-A permits one to instantiate a gimbal module any number of times, and to "connect" them by accessing the power-conjugate variables at their "nodes". The Verilog-A simulator automatically accounts for the interrelation of the connected variables. For example, the inertia matrix expressed in the inertial frame can be written as Trans([R])*[Io]*[R], where [Io] is the inertia matrix expressed in the body-fixed frame and [R] is the transformation matrix between the inertial frame and the body frame. Another example of the vector/matrix syntax in extended Verilog-A is the formulation of the cross-product matrix using the fully anti-symmetric rank three tensor [x]. In that way, the cross-product of any tensors [A] and [B] can be written simply as [A]*[x]*[B]. Thus, it is convenient to write:

$$[Io] = [Ic] - [m]*[rc]*[x]*[rc]*[x] \qquad (27)$$

Refer to Figure 2 for the notational reference of the gimbal frame relations being implemented in the Verilog-A code given in the Appendix. In this Verilog-A model, g_out and g are conservative nodes with a "rotational_omega" discipline. In the standard header of Verilog-A, appearing in the top line of the code, the rotational_omega discipline is defined, and it has the

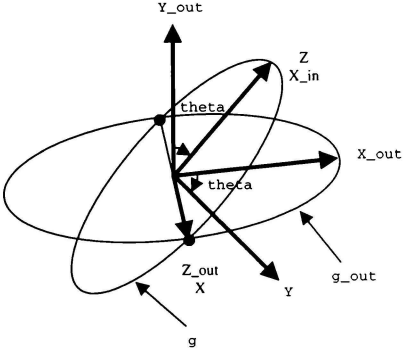


Figure 2: Gimbal Notation g_out and g

potential access function "Omega" for angular velocity, and the "Tau" flow access function for torque. Note that all of the equations with access functions on the left-hand-sides are of the *contribution-statement* form (i.e., using "<+"); this means that many modules can contribute to the

access function of a node, since one node may be connected to many modules. The simulator determines the value of the access function of a node globally after accounting for all of the local contributions from the different modules and solving the resulting system of equations. For example, the g corresponding to each gimbal receives two contributions to its torque: one from the inner gimbal instance (the reaction torque) and one from the outer gimbal instance (the applied torque). These quantities are analogous to the τ_i, $R^i_{i+1}\tau_{i+1}$ of Eq. (5). All other nodes are non-conservative and are used to carry the values of the kinematic unit vectors.

IV. Open-Loop Response of a 3-Gimbal System

Here we present simulation results for a 3-gimbal system in an open-loop mode. General case mass properties are given whose units are consistent with those of the torques; thus, the angular accelerations and angular velocities will be [rad/sec^2] and [rad/sec], respectively. Plotted separately are the three gimbal rotational axis angular velocities – retrieved from Eq. (2) as a function of the angular momentum states. Please note that all notation is consistent with the body-fixed frame development of Section II; thus, the inertial frame velocities from the Verilog-A model were projected onto the moving gimbal rotational axes for consistency. For distinction and clarity, the responses as contributed from torque motor forcing functions and base frame angular and linear motion are here presented separately. What is being shown are the consistent results from all implementations (Matlab^{TM}, EASY5^{TM}, and the Verilog-A simulator).

Gimbal	Ixx	Ixy	Ixz	Iyy	Iyz	Izz
Outer	.05	.005	.001	.07	-.001	.06
Middle	.044	-.004	.002	.045	.015	.046
Inner	.077	.007	.002	.065	-.001	.09

Table 1: Gimbal Centroidal Inertia Matrices I_i^c

Gimbal	m	r_x^c	r_y^c	r_z^c
Outer	10	-.001	.002	-.003
Middle	10	.002	-.0015	.001
Inner	35	.003	.001	.001

Table 2: Gimbal Mass and r_i^c vectors

A. Torque-motor Response

$T_i^N = [.3\cos(2\pi)t \;\; .5\cos(1.6\pi)t \;\; .7\cos(.8\pi)t]$

$\dot{\omega}_{I0}, \omega_{I0}, a_{I0} = [0\;\;0\;\;0]$

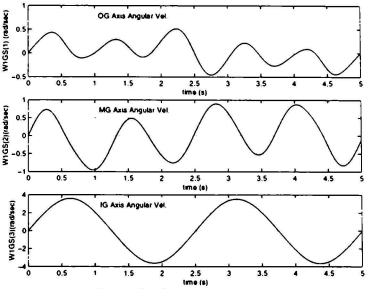


Figure 3: Torque Motor Response

B. Angular Base Motion Response

$T_i^N = [0\;\;0\;\;0]$

$\omega_{I0} = [3\;\;-5\;\;7]$

$\dot{\omega}_{I0}, a_{I0} = [0\;\;0\;\;0]$

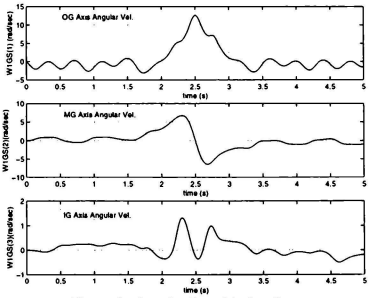


Figure 4: Angular Base Motion Response

C. Linear Base Motion Response

$T_i^N, \dot{\omega}_{I0}, \omega_{I0} = [0\;\;0\;\;0]$

$a_{I0} = [40\;\;60\;\;80]$

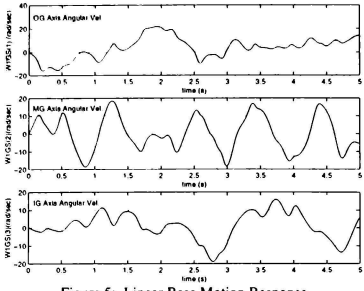


Figure 5: Linear Base Motion Response

V. Discussion

One motive for implementing the explicit formulation of the gimbal dynamic problem in two of the popular control systems simulation tools was verification of the Verilog-A model. Another motive anticipates the application: the gyro-controlled platform, pointing control of a camera, etc. The use of the gimbal system as a plant model substantially broadens the discussion, since one now must contend with the issues of user-flexibility in digital control design, and larger scale mixed analog-digital system analysis. With respect to the INS application, having successfully controlled the plant to meet various requirements, it is tempting to add additional guidance or navigation functionality and so forth. The Fortran-based EASY5™ tool renders matrix/vector operations unattractive at best, and object-orientation is lacking; and for reasonable execution speeds, one must further produce executable code from the Matlab™ model. On the other hand, still it must be remarked that implicit modeling capabilities are available in most of the commercial control systems simulation packages (to differing degrees). So one may easily reduce the complexity of the explicit formulation described in Section II by defining the constraint torques as implicit states and retrieve much of the elegance and modularity touted by the Verilog-A model. But one important distinction is that however the plant dynamics are implemented, a good deal of information is available in the symbolic equations of motion which will be of interest to the control analyst. The functional dependence of the gimbal axis coupling on configuration angles, for example, or the analytical form of the nonlinear "disturbance terms". These are not at all visible in the Verilog-A model, nor would they directly appear to be in any similar implicit formulation.

It has been remarked[5,6] that object-oriented modeling environments for physical systems should possess, among other attributes: topological interconnection capability, de-bugging capabilities, hierarchical modeling, class inheritance, object instantiation, and generalized networking capability, to name a few. In part, Verilog-A responds to these calls, as do other environments. Class inheritance is still lacking, and this is strongly motivated by the desire for smooth de-bugging capability.

VI. References

[1] Hughes, P. C., *Spacecraft Attitude Dynamics*, 1st Ed., Wiley, 1986, pp. 56-58.

[2] Shabana, A. A., *Computational Dynamics*, Wiley, 1994, pp. 416-432.

[3] Craig, J. J., *Introduction to Robotics*, 2nd Ed., Addison-Wesley, 1989, pp. 195-201.

[4] Verilog-AMS Language Reference Manual: Analog and Mixed Signal Extensions to Verilog HDL, Version 1.3, Open Verilog International, 1999.

[5] Cellier, F. E., et. Al., "Modeling from Physical Principles", in *The Control Handbook*, CRC Press, 1996, pp. 99-108.

[6] Otter, M., and Cellier, F. E., "Software for Modeling and Simulating Control Systems", appearing in *The Control Handbook*, CRC Press, 1996, pp. 415-428.

Appendix: Verilog-A Gimbal Model

```
module gimbal(g_out, g, Z_out, Y_out, Z, Y, a);
   /* Node declarations */
   input          [1:3] g_out, Z_out, Y_out, a;
   output         [1:3] g, Z, Y;
   rotational_omega [1:3] g_out, g;
   position       [1:3] Z_out, Y_out, Z, Y;
   acceleration   [1:3] a;
   /* Parameter declarations */
   parameter real To=1, Ko=1, m=0, w_motor_0=0;
   parameter real Ic[1:3][1:3]={1,0,0, 0,1,0, 0,0,1};
   parameter real rc[1:3]={0,0,0}, K_joint=1000;
   /* Local variable declarations */
   real X_out[1:3], w_motor, theta, P[1:3][1:3];
   real x[1:3][1:3][1:3], P_YZ[1:3][1:3], Io[1:3][1:3];
   analog begin
      /* Kinematics */
      [X_out]=[Pos(Y_out)]*[x]*[Pos(Z_out)];
      [Pos(Z)] <+ [Pos(Y_out)]*(cos(theta)) +
                                    [X_out]*(sin(theta));
      [Pos(Y)] <+ [Pos(Z)]*[x]*[Pos(Z_out)];
      [P[1]]=[Pos(Z_out)]; [P[2]]=[Pos(Y)];
      [P[3]]=[Pos(Z)];
      w_motor=[[Omega(g,g_out)]*[Pos(Z_out)]][1];
      theta=-idt(w_motor,0.);
      /* Dynamics */
      if(analysis("ic")) begin      /* Initialization */
        [Io]=[Ic]-(m)*[rc]*[x]*[rc]*[x];
        [x]=(0,0,0,0,0,-1,0,1,0,0,0,1,0,0,0,
                              -1,0,0,0,0,-1,0,1,0,0,0,0,0);
        [P_YZ]=(0,0,0, 0,1,0, 0,0,1);
        [Omega(g,g_out)] <+ (w_motor_0)*[Pos(Z_out)];
      end else begin                /* Main dynamics */
        [Tau(g_out,g)] <+
            (To*cos(2*`M_PI*Ko*$realtime))*[Pos(Z_out)]
    + (K_joint)*Trans([P])*[P_YZ]*[P]*[Omega(g_out,g)];
        [Tau(g)] <+ ddt(Trans([P])*[Io]*[P]*[Omega(g)])
                    + (m)*Trans([P])*[rc]*[x]*[Acc(a)];
      end
   end
endmodule
```

INVESTIGATION OF DYNAMIC STRUCTURAL MODELS SUITABLE FOR THE SIMULATION OF LARGE AIRCRAFT

Wendy N. Hashii*, Graduate Student
Aaron R. Munger†, Graduate Student
Eltahry Elghandour‡, Ph.D.
Faysal Kolkailah§, Ph.D.
Daniel J. Biezad¶, Ph.D.

California Polytechnic State University, San Luis Obispo, California 93407

Abstract

The simulation of modern aircraft with lightweight structures and high bandwidth control systems requires more sophisticated modeling than the standard six degree-of-freedom representation. The objective in this work is to investigate and test methods for incorporating longitudinal structural mode vibration data into the simulation equations and to allow for variation in frequency and shape functions relative to an established baseline. Using a two-dimensional COSMOS/M finite element model, the thickness and the length of the aircraft model were varied to match the resonant frequency of the baseline model. The addition of forces then matched the mode shape of the computer model to the actual shape of the baseline. The results, confirmed by experimental tests, indicate that it is possible to effectively model the natural frequency and mode shape at any longitudinal location as center of gravity and/or weight is varied about a nominal condition. Moreover, the methodology allows new configuration baselines to be established by the matching of modal frequencies and shapes to external data from ground or flight tests.

Introduction

The strength and flexibility characteristics of large, modern aircraft structures often produce structural modes of vibration that are of the same order of magnitude as the bare airframe short-period response. For example, YF-12 pilots have occasionally reported longitudinal pilot-induced oscillation (PIO) tendencies during refueling operations caused by the interaction of the pilot with a combination of the aircraft's short-period poles and the structural first bending mode.[1] The first-order structural mode may in this case have an effect on the handling qualities of the aircraft that should be taken into account in a piloted simulation of the vehicle. Longitudinal control system designs, such as that considered for a linearized dynamic model of a supersonic transport aircraft, are often characterized by excessive overshoot and large settling time for large, high-speed aircraft without the inclusion of aeroelastic modes in the controller.[2]

The background literature[3-9] available on flight simulation models shows that it is acceptable to assume that the non-linear simulation equations may be augmented with linear structural vibration equations to adequately describe the resulting motion of large aircraft as seen by the pilot. The effects of structural vibrations have also been successively simulated using a lumped-mass dynamic model[10] and aircraft loads prediction has been achieved through the use of finite element aerodynamics.[11-12] However, these models are complex and difficult to implement in piloted simulations and are not easily adapted to experimental and test data.

With the aid of COSMOS/M, a finite element (FE) analysis program that runs on a personal computer, the mode frequencies and shapes of different aircraft configurations may be simulated and evaluated. This work describes the fabrication and testing of a scalable model that generates the longitudinal modal frequencies and shapes of actual aircraft. Results from matching experimental and finite element predictions

* Aeronautical Engineering Master's Student, Member, AIAA
† Aeronautical Engineering Master's Student, Member, AIAA
‡ Assistant Professor, Department of Aeronautical Engineering
§ Professor, Department of Aeronautical Engineering
¶ Professor, Department of Aeronautical Engineering, Associate Fellow, AIAA
Copyright © 1999 by W. Hashii. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

justify continuing this approach for generic aircraft models.

The general idea was to determine the simplest FE model and modeling process that matched the modal frequencies and shape of a given baseline. The process chosen was scalable to large aircraft applications. It was initiated by determining a nominal frequency of an actual aircraft configuration (baseline) to be matched. The thickness of the FE model was then varied so that its resonant vibration frequency matched this value. After the establishment of this initial baseline configuration, the FE analysis was used to find the sensitivity of modal frequencies and shapes to center of gravity (excitation location) and weight (thickness) changes for any sensor location along the longitudinal axis. The resulting shape function determined the effect of longitudinal vibration on the flight simulation equations of motion for the aircraft.

The process also allows for modifying the baseline configuration to match ground or flight vibration data. A significant attempt to combine simulation and structural equations was accomplished by Powers[13] for the YF-12 strategic reconnaissance aircraft. Powers assumed the simple dynamic shape function of a uniform beam for the primary structural mode shape, and defined the parameters to be adapted to flight test data. The uniform beam solution provided a good fit of the actual aircraft deflection (obtained from ground vibration testing) except near the aft end of the aircraft. The purpose of this work is to use a FE technique to accommodate more than one longitudinal mode, to account for the lack of proper fit at the aft limit of the aircraft, and to identify model parameters that can be adapted to test data.

The mode shape, which expresses fuselage deflection as a function of fuselage location, can be used to develop transfer functions suitable for flight simulation.[14] Deflections from the dynamic equations that describe this mode shape can be added to the rigid body motion to obtain the total deflections of the aircraft structure.

Procedure

Experimental Test Model

The 2-dimensional representation of the aircraft was tested and fabricated to correspond to the scaled dimensions of the Boeing 747 as shown in Figure 1. A spring was added to simulate the vibration of the aircraft. Constructed from 6061 Aluminum (E = 10.1 x 10^6 psi), this 22-inch flat plate model was connected to the shaker table by a screw (see Figure 2). The shaker table was connected to an amplifier that was connected to a function generator, which produced a sine wave. By varying the frequency and amplitude of the sine wave, the fundamental bending mode and second bending mode were discovered and recorded.

To identify the resonant lines of the test specimen, the plate was mounted on the shaking table and a function generator was used to scan through the frequencies ranging from 0 to 500 Hz. Resonant frequencies were identified by sharp rises in audible amplitudes, visual plate displacements, and the formation of node lines using sugar traces. In order to pinpoint the first two bending modes, small amounts of sugar were poured on the plane. When the sugar granules formed straight lines, the locations of "zero displacement" on the wing could be detected. By counting the number of locations with "zero

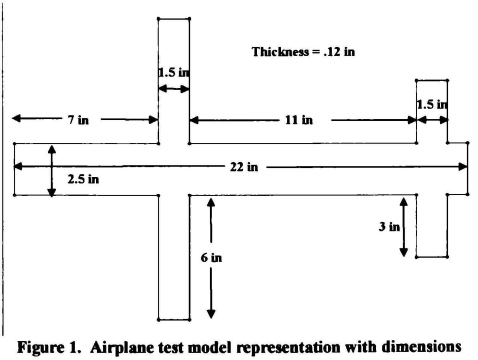


Figure 1. Airplane test model representation with dimensions

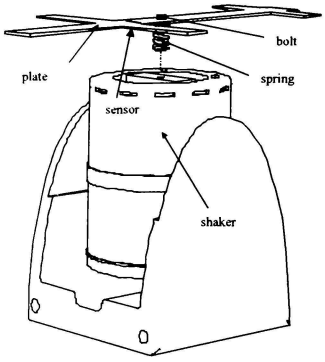


Figure 2. Airplane test model and shaker setup

displacement" and noting their locations relative to the fixed screw in the center of the plane, the bending modes were discovered. A schematic of the experimental setup for dynamic testing of the plate model is presented in Figure 3.

The experimental results show that the observed vibration frequency and mode shape for the first longitudinal mode match the finite element model very closely. A comparison of the predicted COSMOS/M natural frequency to the experimental natural frequency was conducted with the aid of LabVIEW software. A ceramic sensor was attached in order to obtain the frequency response of the model to random excitations. Figure 4 shows the results of this investigation. Each peak corresponds to a different natural frequency response of the model. The experimental results predict a natural frequency of the first mode to be approximately 50 Hz. These experimental results match the COSMOS/M predicted value of 52 Hz within 4%. This 4% error was attributed to experimental error in the function generator.

Finite Element Analysis

Most of the data presented here came from a finite element analysis of various configurations of the original 2-D aircraft model. COSMOS/M was the program used to perform the analysis. The model was constructed in two parts. One part was the plane representation and the other was the spring representation. Using the same dimensions as the test model, the first part (plane model) was constructed from points and surfaces in COSMOS/M. The resulting

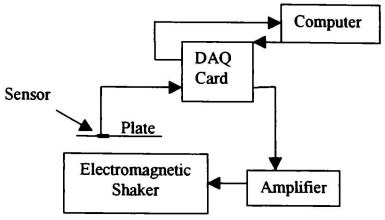


Figure 3. Schematic of dynamic testing

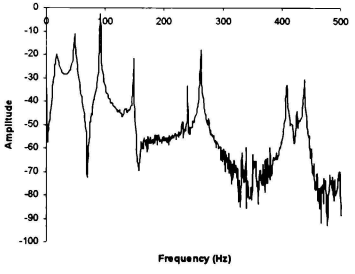


Figure 4. Experimental results

COSMOS/M drawing was modeled as a 2-D four-node shell surface. The material was specified as aluminum 6061, the same material as the experimental model. In order to represent the experimental model, the thickness was specified 0.12 inches. The next step was to mesh the surface with elements. Eighty-eight elements were added along the length of the fuselage, and four elements were added along the width. Six elements were added along the lengths of the wing and six elements were added along the widths. Three elements were added along the lengths of the tail and six were added along the widths of the tail. A numerical study conducted by varying the number of elements determined that the number of elements specified yielded accurate results. For the second part of the model, the spring representation was added. A curve was added to the center of gravity location of the model. The curve was specified as a spring element

composed of steel alloy, with a spring constant of 50. The real constants were left at default settings (i.e. the spring was given only one degree of freedom). The spring was meshed as one element with two nodes. Finally, the nodes were merged and compressed and the constraints (boundary conditions) were added. Figure 5 shows the resulting COSMOS/M model. A frequency analysis was done, and the first and second bending modes were found by observing the resulting deflection.

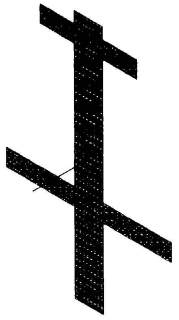


Figure 5. Finite element model

Modeling and Simulation Process

Establishing Baseline from GVT

The most significant structural dynamic response of the aircraft relative to handling qualities is the longitudinal first bending mode. Piloted simulations seldom include the structural modes because of the complexity involved. The purpose of this study was to develop an easy method to model the structural modes of large aircraft that can be used in a pilot simulation and calibrated by flight data.

To be implemented into a piloted simulation, the accuracy of the shape function is vital not only for the motion at the pilot station, but also for modeling the effects of varying locations for sensors and actuators. Reference 13 models ground vibration test (GVT) data obtained for the YF-12 aircraft. In this study, a curve fir for the data was obtained using the sinusoidal of the uniform beam solution. This solution was found to provide a good fit except at the aft ends of the model.

To establish a baseline that can be modified by weight and center of gravity location about a nominal condition, a generic FE element model was created. In this case, the aircraft was treated as a combination of a fuselage (including the tail) and left and right wings.

In order to demonstrate the process used for

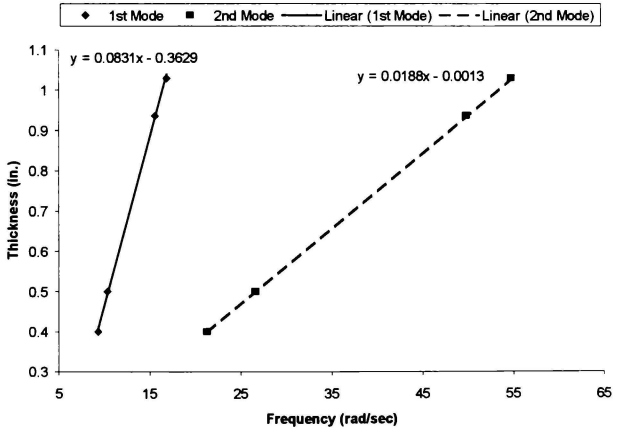


Figure 6. Frequency determination by model thickness

matching GVT data, the frequency and first structural bending mode of the YF-12 aircraft[1] is matched as described here. The first step is to match the frequency of the structural vibrations. By changing the both the length and the thickness of the FE model, the frequency from the GVT can be matched. Figure 6 shows the thickness of the model as a function of frequency for both the first and the second bending modes. From Figure 6, the thickness required to match the 15.7 rad/s frequency of the YF-12 was 0.936 in. with a model length of 22 ft. However, there is a limit to the range of frequencies for this graph. To achieve a frequency beyond the range of this graph, the length of the model can be increased or decreased to give a lower or higher frequency, respectively.

Once the frequency of the FE model has been defined, the structural bending mode can be matched by applying forces to various points on the fuselage. The use of a constant force applied to the fuselage does not alter the frequency of the model. Therefore, the mode shape can be adjusted without iterating to find the frequency. By applying forces to the nose, to the aft end, and around the center of gravity location on the fuselage, changing the magnitude of the forces can alter the mode shape. Using the relationship between the deflection and the magnitude of the forces, the magnitude of the forces required to match the mode shape can be determined for the parameters (length and thickness) of a given model.

Figure 7 shows a side view of the placement of the forces on the FE model and the resulting mode shape. The forces applied to the nose, aft end, and near the center of gravity are constant forces. The force

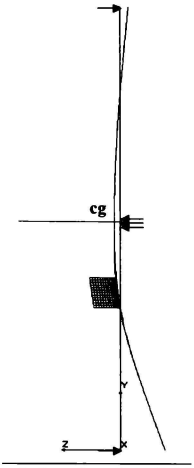


Figure 7. Location of applied forces

applied at the center of gravity provides a harmonic force to simulate the vibration of the aircraft in a time history analysis. This force takes the place of the spring used in the frequency analysis. Figure 8 shows the relationship between the deflection of the model as a function of the magnitude of a force applied at the

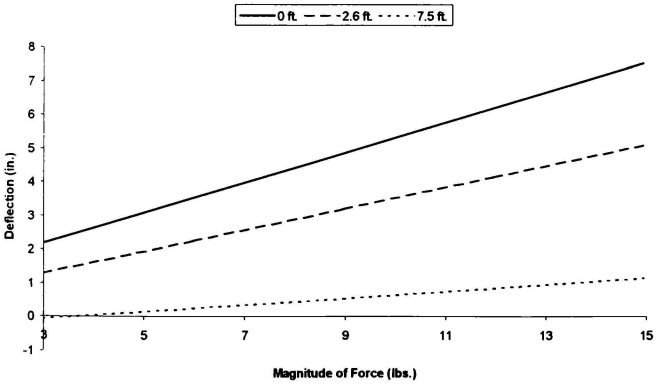


Figure 8. Changing magnitude of forces to match mode shape

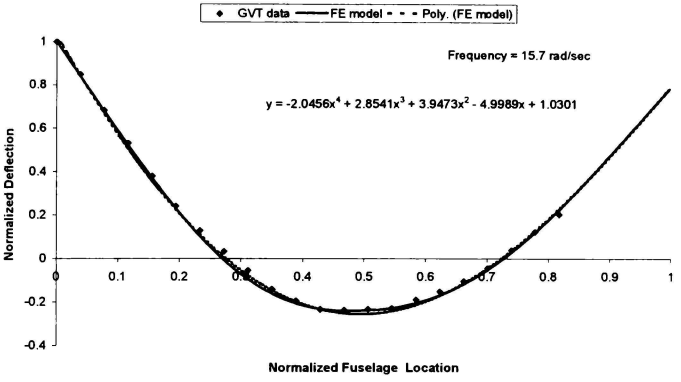


Figure 9. Polynomial curve fit of FE model

nose of the fuselage. The three curves on the graph refer to three different locations along longitudinal direction of the fuselage.

Using a series of curves relating the magnitude of the deflection with the associated magnitude of the applied force, the mode shape can be found by iterating the forces for a required deflection. Note that a different set of curves is required for each variation in length and thickness. The resulting match between the FE model and the GVT data (as read from Reference 13) is shown in Figure 9. From the plot of the deflection, a polynomial equation was determined by curve fitting the data. This equation can be implemented into a piloted flight simulation to provide the deflection as a function of fuselage location.

The second structural bending mode requires a more complex process in order to provide the necessary data for matching purposes. The frequency can be matched using the same method as for the first bending mode (see Figure 6). However, COSMOS/M cannot determine the deflection using the addition of forces for the second bending mode with the time history analysis method. To modify the mode shape for the second mode, additional mass can be added to the FE model. However, the addition of mass changes the frequency creating an iterative process between matching the frequency and matching the deflection.

Modeling Changes in Weight and Center of Gravity Location

One of the main issues in providing simulation support for flight research is that the structural modes available from either a prediction or a GVT often do not correspond to the configuration currently being tested. Because of cost and time constraints, it may not be feasible to update the structural dynamic models for each configuration. As a result, a simple method that can predict the changes in the structural model due to changes in the weight or center of gravity (cg) location of a baseline aircraft configuration has been developed.

Changes in the cg location around a baseline configuration can be predicted by changing the excitation location of the FE model. Figure 10 shows the changes in the mode shape as the excitation location is varied. The curves, representing three different cg locations, in Figure 10 are all normalized to the same value so the curves can be compared and contrasted. As shown, moving the cg location aft decreases the deflection at the nose and increases the deflection at the tail.

Simulating changes in the weight or weight distribution of an aircraft can be accomplished by adding mass to the FE model. Figure 11 shows a FE model with mass added. Adding a lumped mass to the FE model reduces the frequency by

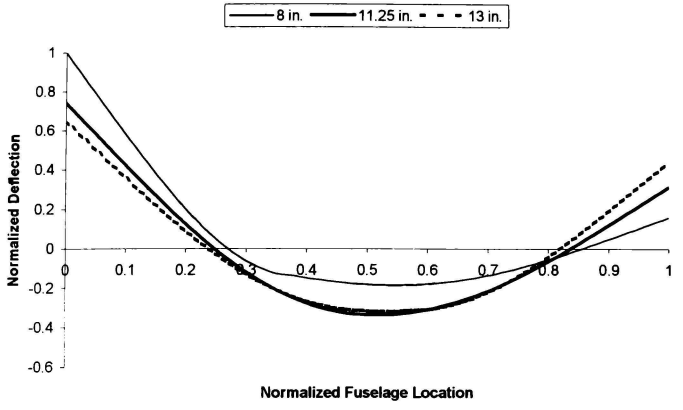


Figure 10. Mode shape for various excitation locations

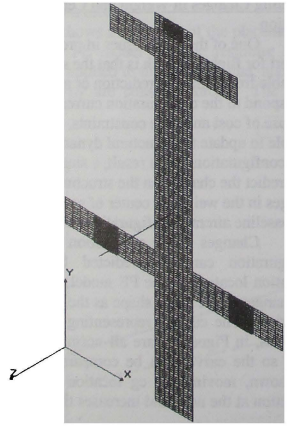


Figure 11. FE model with mass addition

damping the vibrations while also deflecting the mode shape.

Future Considerations: Implementation of Active Control

After modeling and simulating the primary structural bending modes, the next stage is to determine a viable method to control the structure. For this purpose, a flexible model aircraft approximately 11 feet in length has been constructed to demonstrate the significant structural bending effects of an aircraft in flight. The fuselage of the model aircraft is divided into two parts. Constructed of wood, the first part of the fuselage supports the wings, the batteries, and the motor. The second part of the fuselage is constructed of a highly flexible Aluminum tube. The model aircraft was constructed in two parts in order to demonstrate the impact of the structural bending modes while still allowing the aircraft to attain flight.

With the absence of GVT data, a COSMOS/M FE model that will accurately predict both the frequency and the structural bending modes is currently being developed. Figure 12 shows a preliminary version of the FE model. Methods to simulate the bending effects of the wooden fuselage section and the wings (composed of foam core, carbon fiber stiffeners, and fiberglass) as well as the damping effect due to the weight of the batteries and motor are currently being investigated. Once a baseline frequency and mode

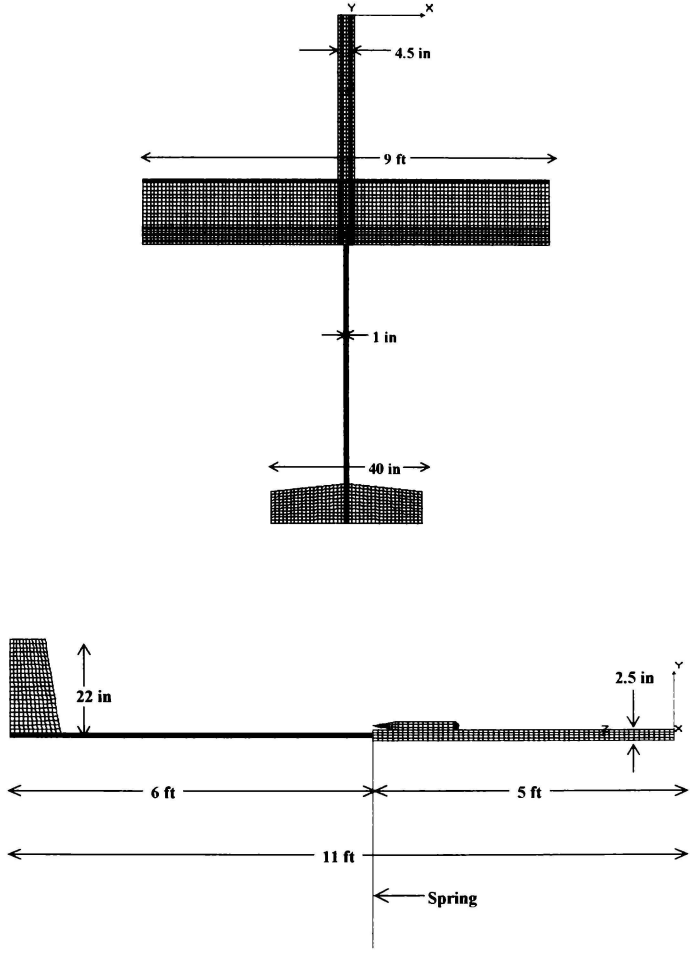


Figure 12. Preliminary FE model of model aircraft

shape have been established for the model aircraft, the process outlined above can be used to predict the effect of changing the cg location or the weight of the aircraft.

Next, the shape function, which expresses fuselage deflection as a function of fuselage location, can be used to develop transfer functions suitable for flight simulation. Deflections from the dynamic equations that describe this mode shape are added to the rigid body motion to obtain the total deflections of the aircraft structure. The final stage will be to apply active control to damp, isolate, or control vibration. Through the use of sensors to measure vibrations, active control electronics can be used to apply feedback algorithms, and strain actuators to induce "anti-vibrations" to counteract the structural vibrations.

Concluding Remarks

Using these results, the frequency and mode shape of the first natural bending mode of ground vibration test data may be modeled by changing both the thickness and the length of the model and by using forces or mass to change the structural bending mode of the FE model. First, the natural frequency can be matched by adjusting the thickness according to a linear relationship. Then, the first structural bending mode can be matched using applied forces to deflect the shape. Altering the thickness, length, and mass of the FE model can approximate the second structural bending mode. A mathematical method to model a given frequency and mode shape for both the first and the second structural bending modes is in progress.

For flight simulation, the relationships between the structural and the mean axis and between the deflected fuselage and the mean axis for a variety of flight station locations along the longitudinal axis must be known. From these relationships, the structural bending modes can be implemented into piloted simulations.

Acknowledgment

The authors are grateful to Mr. Fred Webster and Mr. Kurt Buehler of the Air Force Flight Test Center, Edwards Air Force Base, and to Dr. Len Sakell of the U.S. Air Force Office of Scientific Research, for their support of this research project.

References

1. Smith, J. W., Berry, Donald T., "Analysis of Longitudinal Pilot-Induced Oscillation Tendencies of YF-12 Aircraft," NASA TN D-7900, February 1975.

2. Joshi, S. M., Kelkar, A. G., "Inner Loop Control of Supersonic Aircraft in the Presence of Aeroelastic Modes," IEEE Transactions on Control Systems Technology, Vol. 6, No. 6, Nov. 1998, pages 730-739.

3. Shafer, M.F.,"Low-Order Equivalent Models of Highly Augmented Aircraft Determined from Flight Data," AIAA Journal of Guidance, Control, and Dynamics, Sept 1982.

4. Waszak, M. R., and D. K. Schmidt, "Flight Dynamics of Aeroelastic Vehicles," AIAA Journal of Aircraft, Vol. 25, No. 6, June 1988 , pages 563-571.

5. Newman, B., and D. K. Schmidt, "Numerical and Literal Aeroelastic Vehicle Model Reduction for Feedback Control Synthesis," AIAA Journal of Guidance, Control, and Dynamics, Vol. 14, No. 5, pages 943-953, 1991.

6. Waszak, M. R., Buttrill, C. S., and D. K. Schmidt, "Modeling and Model Simplification of Aeroelastic Vehicles: An Overview," NASA TM 107691, Sept. 1992.

7. Meirovitch, L., "Hybrid State Equations of Motion for Flexible Bodies in Terms of Quasi-Coordinates," AIAA Journal of Guidance, Control, and Dynamics, Vol. 14, No. 5, Sep.-Oct. 1991, pages 1008-1013.

8. Newman, B., and D. K. Schmidt, "Aeroelastic 'Vehicle Multivariable Control Synthesis with Analytical Robustness Evaluation," AIAA Journal of Guidance, Control, and Dynamics, Vol. 17, No. 6, Nov.-Dec. 1994, pages 1145-1153.

9. Livneh, R., and D. K. Schmidt, "Unified Literal Approximations for Longitudinal Dynamics of Flexible Flight Vehicles," AIAA Journal of Guidance, Control, and Dynamics, Vol 18, No. 5, Sept-Oct. 1995, pages 1074-1083.

10. Chavez, F. R., and D. K. Schmidt, "Analytical Aeropropulsive/Aeroelastic Hypersonic-Vehicle Model with Dynamic Analysis," AIAA Journal of Guidance, Control, and Dynamics, Vol. 17, No. 6, Nov-Dec. 1994, pages 1308-1319.

11. Rowan, J. C., and Burns, T. A., "Aeroelastic Loads Predictions using Finite Element Aerodynamics," Journal of Aircraft, Vol. 12, No. 11, Nov. 1975, pages 890-898.

12. Cunningham, H. J., Batina, J., T., and Bennett, R. M., "Modern Wing Flutter Analysis by Computational Fluid Dynamics Methods," Journal of Aircraft, Vol. 25, No. 10, Oct. 1988, pages 962-967.

13. Powers, B. G.,"Structural Dynamic Model Obtained from Flight for Use with Piloted Simulation and Handling Qualities Analysis." NASA TM 4747, 1996.

14. Munger, A.R., et al, "Transfer Functions for Simulating the Structural Vibration Modes of Large Aircraft," in Proc. AIAA AFM Conf., Portland, OR., Paper AIAA-99-4108, Aug. 1999.

15. Anderson, M.R., "Robustness Evaluation of a Flexible Aircraft Control System," AIAA Journal of Guidance, Control, and Dynamics, Vol. 16, No. 3, May-Jun 1993, pages 564-571.

16. Chen, C.W., Huang, J.K., Phan, M., Juang, J.N.,"Integrated System Identification and State Estimation for Control of Flexible Space Structures," AIAA Journal of Guidance, Control, and Dynamics, Vol. 15, No. 1, Jan-Feb 1992, pages 88-95.

17. Choe, K., Baruh, H.,"Actuator Placement in Structural Control," AIAA Journal of Guidance, Control, and Dynamics, Vol. 15, No. 1, Jan-Feb 1992, pages 40-48.

18. Lim, K.B., "Method for Optimal Actuator and Sensor Placement for Large Flexible Structures," AIAA Journal of Guidance, Control, and Dynamics, Vol. 15, No. 1, Jan-Feb 1992, pages 49-57.

19. Gilbert, M. G., Schmidt, D. K.,"Integrated Structure/Control Law Design by Multilevel Optimization," AIAA Journal of Guidance, Control, and Dynamics, Vol. 14, No. 5, Sept.-Oct. 1991, pages 1001-1007.

20. Meirovitch, L., "Hybrid State Equations of Motion for Flexible Bodies in Terms of Quasi-Coordinates," AIAA Journal of Guidance, Control, and Dynamics, Vol. 14, No. 5, Sept.-Oct. 1991, pages 1008-1013.

21. An, J., Yan, M., Zhou, W, Sun, X., Yan, Z, Qiu, C.,"Aircraft Dynamic Response to Variable Wing Sweep Geometry," AIAA Journal of Aircraft, Vol. 25, No. 3, March 1988 , pages 216-221.

22. Lottati, I., "Aeroelastic Tailoring of a Composite Wing with a Decoupler Pylon as a Wing/Store Flutter Suppressor," AIAA Journal of Aircraft, Vol. 25, No. 3, March 1988 , pages 271-280.

23. Karpel, M.,"Efficient Vibration Mode Analysis of Aircraft with Multiple External Store Configurations," AIAA Journal of Aircraft, Vol. 25, No. 8, August 1988 , pages 747-751.

24. Gilbert, M. G., Schmidt, D. K., Weisshaar, T.A.,"Quadratic Synthesis of Integrated Active Controls for an Aeroelastic Forward-Swept-Wing Aircraft," AIAA Journal of Guidance, Control, and Dynamics, Vol. 7, No. 2, Mar-Apr 1984, pages 190-196.

25. Gawronshi, W., Williams, T.,"Model Reduction for Flexible Space Structures." AIAA Journal of Guidance, Control, and Dynamics, Vol. 14, No. 1, Jan-Feb. 1991, pages 68-75.

26. Williams, T.," Closed-Form Grammians and Model Reduction for Flexible Space Structures." IEEE Transactions on Automatic Control, Vol. 35, No. 3, March 1990, pages 379-382.

FLIGHT SIMULATION AND CONTROL SYSTEM DESIGN SOFTWARE FOR PROTOTPYE DEVELOPMENT

Kapseong Ro*, Mykhaylo Demyanov†, Juwon Park‡, Sangwhee Go§ & Chul-Ho Lim¶

Korea Aerospace Research Institute (KARI)
P.O. Box 113, Yusong, Taejon, 305-600, Korea

ABSTRACT

Flight simulation and control system design software to be used for prototype development is presented. This software is developed in MATLAB®/ SIMULINK® environment in that all inputs and outputs of complex aircraft motion and its systems are represented into simple block diagram models. The software includes typical linear models of various orders as well as the full six degree-of-freedom nonlinear equations of motion for full flight envelope including the motion on the ground. Aircraft subsystems such as propulsion system, flight control system, flight instruments, landing gears, etc. are also represented as graphical SIMULINK® blocks. Using these blocks with standard libraries of MATLAB®/ SIMULINK®, the software can be used to design and test the stability and control augmentation system (SCAS) using various methodologies, as well as to develop an engineering simulator for the prototype development. Also presented are some iterative numerical procedures to compute initial conditions for practical cases of steady airplane motion in order to develop corresponding linear models. An easy to use graphical user interface (GUI) is designed that allows simply organizing the procedures of simulation tasks, visualizing and analyzing the simulation results.

NOTATIONS

$F_N, F_L, F_V, F_B, F_W, F_A$: Inertial, Earth-surface, Vehicle-carried-vertical, Body-fixed, Stability, Wind and Atmosphere-fixed reference frames, respectively, as in reference [1].

$\vec{R}^{XY}$: Position vector from a point X to Y.

${}^{Y}\vec{V}^{X}$: Velocity of point X in a reference frame F_Y

${}^{Y}\vec{\omega}^{X}$: Angular velocity of a reference frame F_X in a reference frame F_Y.

$\vec{\vec{X}}$: A dyadic.

$\vec{X}_Y, \vec{\vec{X}}_Y$: Components expression of a vector and a dyadic of $\vec{X}$ and $\vec{\vec{X}}$, respectively, in the Cartesian coordinates of reference frame F_Y, i.e.

$$\vec{X}_Y = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \vec{\vec{X}}_Y = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

$\tilde{\vec{X}}_Y$: Skew symmetric matrix from of $\vec{X}_Y$, i.e.

$$\tilde{\vec{X}}_Y = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

L_{XY} : Transformation matrix from a reference frame F_Y to a reference frame F_X.

INTRODUCTION

Flight simulation is an essential tool in the design and development of a prototype and its systems. The full set of nonlinear six degree-of-freedom (DOF) aircraft equations of motion should be solved, that requires to develop highly complex, computationally intensive computer code including the integration of differential equation, multidimensional table look-up, solving a set of non-linear simultaneous equations, etc. It is also multi-disciplinary in its nature, so that the integration of the design results from each discipline into an appropriate format for flight simulation has been a considerable task. In this paper, a simple and graphical approach to carry out flight simulation and control system design is proposed using a commercial software package, MATLAB®/ SIMULINK®[2]. A set of SIMULINK® block

---

*. Senior Researcher, Aircraft Division
†. Visiting Scholar, Aircraft Division
‡. Researcher, Aircraft Division
§. Researcher, Aircraft Division
¶. Director of Aircraft Division

Copyright © 1999 by KARI. Published by the American Institute of Aeronautics and Astronautics Inc. with permission.

libraries are developed, which represents the full 6 DOF nonlinear equations of motion with independent blocks for aircraft systems of propulsion, flight controls, flight instruments, landing gears, etc. A nonlinear model representing only longitudinal motion is also included in the block libraries, as well as typical linear approximations at practical steady state motion. Using these blocks with standard MATLAB®/SIMULINK® blocks and libraries, it provides a simple and easy to use, but very powerful design tool for a prototype aircraft development. It can be used to design and test the flight instrument as in reference [3]. In particular, the comprehensive features of MATLAB® and its toolboxes provide great flexibility and productivity to a flight control system designer. Furthermore, the developed models are ready to be implemented into an engineering flight simulator. In conjunction with SIMULINK® graphical user interface additional user friendly GUI is developed for use with these libraries.

DESCRIPTIONS OF MODEL LIBRARIES

In the process of prototype aircraft development, a set of linear and nonlinear models of aircraft motion are used to study stability and control characteristics. Both linear and nonlinear model are equally of importance. The linear models are usually used for an analytical design of flight control system structure. The nonlinear models are usually used for more precise estimation of stability and control characteristics of the airplane to account for various nonlinear characteristics, linear model error, etc. These nonlinear models are specially useful for special flight regimes such as high angle of attack and rapid maneuver, where the linear models can not adequately represent the corresponding motion. However, the linear models can quite closely approximate the aircraft motion for much of the flight envelope. Table 1 shows the developed aircraft models represented in SIMULINK® library blocks.

Nonlinear Models : NM-6D, NMP-3D

For the nonlinear 6 DOF model, the main programs are coded in C programming language and then converted to a SIMULINK® *S*-function block to minimize the simulation execution time. The general nonlinear aircraft equations of motion of atmospheric flight on the spherical, rotating earth can be written in vector differential equations are

• Translational Motion :

$$ {}^{F}\dot{\vec{V}}_{B}^{B^*} = -\left({}^{N}\tilde{\vec{\omega}}_{B}^{B} + {}^{N}\tilde{\vec{\omega}}_{B}^{F}\right){}^{F}\vec{V}_{B}^{B^*} + \frac{1}{m}\left[L_{BW}\, m\vec{g}_{I^*} + L_{BW}\vec{F}_{W}^{Aero} + \vec{F}_{B}^{LG} + \sum_{i=1}^{N_e} L_1^T(\psi_{T_i}) L_2^T(\varphi_{T_i})\vec{T}^i\right] $$

	Block Name	Model Properties
1	NM-6D	Full 6 DOF Nonlinear Model w/ Motion on the Ground
2	NMP-3D	Nonlinear 3 DOF Longitudinal Model w/ Motion on the Ground
3	LMPss-4th	4th order Longitudinal State-Space Model
4	LMPss-sp(ph)-2nd	2nd order Longitudinal Short-Period (Phugoid) State-Space Models
5	LMPtf-Yi/Uj-4th	4th order Longitudinal Transfer Function Model
6	LMPtf-Yi/Uj-2nd_sp(ph)	2nd order Longitudinal Transfer Function Model of Short-Period (Phugoid)
7	LMLss-4th	4th order Lateral State-Space Model
8	LMLss-2nd	2nd order Lateral State-Space Model
9	LMLss-1st	1st order Lateral State-Space Model
10	LMLtf-Yi/Uj-4th	4th order Lateral Transfer Function
11	LMLtf-Yi/Uj-2nd	2nd order Lateral Transfer Function
12	LMLtf-1st	1st order Lateral Transfer Function Model
13	LMLss-3nd	3rd order Lateral State-Space Model
14	LMLtf-Yi/Uj-3rd	3rd order Lateral Transfer Function

Table 1. Aircraft Model Blocks

• Rotational Motion :

$$ {}^{N}\dot{\vec{\omega}}_{B}^{B} = -\left(\vec{I}_{B}^{B^*}\right)^{-1}{}^{N}\tilde{\vec{\omega}}_{B}^{B}\vec{I}_{B}^{B^*}{}^{N}\vec{\omega}_{B}^{B} + \left(\vec{I}_{B}^{B^*}\right)^{-1}\left[\vec{M}_{B}^{B^*} + \vec{M}_{LG}^{B^*} + \tilde{\vec{R}}_{B}^{BG^*} L_{BW}\vec{F}_{W}^{Aero} + \sum_{i=1}^{N_e}\tilde{\vec{R}}_{B}^{B^*T_i} L_1^T(\psi_{T^i}) L_2^T(\varphi_{T^i})\vec{T}^i\right] $$

• Attitude Motion (Euler Angles) :

$$ \dot{\mathrm{E}} = R(\mathrm{E})^{I}\vec{\omega}_{B}^{B} $$

• Inertial Position :

$$ \dot{\vec{R}}_{I^*}^{O_E B^*} = L_{IH}{}^{E}\vec{V}_{B}^{B^*} $$

where

B^* : Center of gravity (CG)

$\psi_{T^i}, \varphi_{T^i}$: Incidence angles of ith engine in horizontal and vertical planes, respectively.

$\vec{T}^i$: Thrust vector of ith engine.

L_1, L_2, L_3 : Transformation matrices using Euler angles definition.

$$R(E) = \begin{bmatrix} 1 & \sin\Phi\tan\Theta & \cos\Phi\tan\Theta \\ 0 & \cos\Phi & -\sin\Phi \\ 0 & \sin\Phi\sec\Theta & \cos\Phi\cos\Theta \end{bmatrix}$$

These differential equations are essentials in carrying out the simulation of aircraft motion in atmospheric flight, and with some additional variables the equations can be written into a state-space form as:

$$\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X}, \mathbf{U}, \mathbf{W}, \mathbf{P})$$

$$\mathbf{Y} = \mathbf{G}(\mathbf{X}, \mathbf{U}, \mathbf{W}, \mathbf{P})$$

where

$\mathbf{X} = [U, V, W, P, Q, \cdots]^T$ - state vector

$\mathbf{U} = [\delta_e, \delta_{th}, \cdots]^T$ - input vector

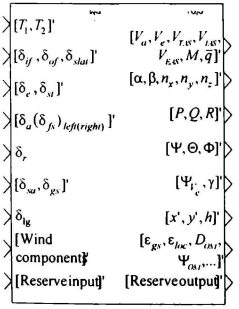
$\mathbf{W} = [\bar{W}_{wind}, \Psi_{wind}, \Gamma_{wind}, \cdots]^T$ - disturbance vector

$\mathbf{P} = [S_{ref}, \bar{c}, \cdots]^T$ - aircraft parameters

The output variables of interest, usually algebraic combination of state variables, often vary depending on a considering task. In case of a full-flight simulator, several hundreds of output variables are necessary. For the developed nonlinear model of **NM-6D** and **NM-3D**, the output variables are chosen to drive necessary flight instruments such as primary flight display (PFD) and navigational display (ND), as well as those variables necessary to design flight control system. Those are,

$$\mathbf{Y} = [V_{TAS}, M, h_{Baro}, h_{radio}, V_z, \Psi, \Theta, \Phi, \beta_{drift}, n_z, \cdots]^T$$

Figure 1 represents **NM-6D** model SIMULINK® library block and its structure.



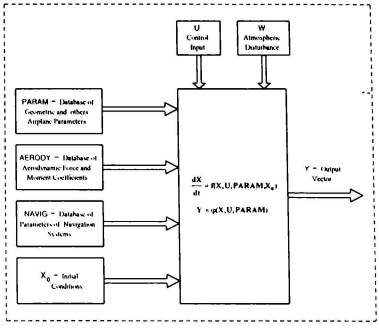


Figure 1. **NM-6D** Library Model & Its Structure

A simplified nonlinear model, **NM-3D**, is obtained by deleting the lateral dynamics from the governing equations of motion, and the output variables are arranged for those representing longitudinal motion of aircraft. Figure 2. shows a SIMULINK® block representing **NMP-3D** model.

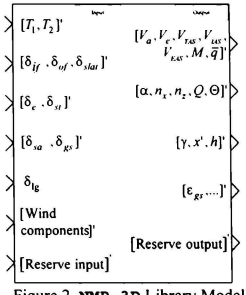


Figure 2. **NMP-3D** Library Model

In order to model the aircraft motion on the runway, landing gear with the ground reaction forces and moments are modeled and programmed in C, then converted into *S*-function to be represented as a SIMULINK® block library model.

Linear Models

Linear models are obtained from the nonlinear 6 DOF model, **NM-6D** with readily available tools in as shown in Figure 2, the main programs are programmed in MATLAB®/SIMULINK® such as **linmod** and **linmod2** function. Before applying these commands, it is necessary to connect **NM-6D** with model inputs and outputs as shown in Figure 3, and by executing the MATLAB® command,

`[A,B,C,D]=linmode('ModelName',X0,U0)`

where '***ModelName***' represents a SIMULINK® block diagram model of Figure 3, `X0` is a vector of values representing for an equilibrium condition of interest, and `U0` is a vector of values for control inputs and disturbance for the corresponding equilibrium condition. Practical cases of equilibrium flight regime and methods to compute corresponding initial conditions are discussed in the following section.

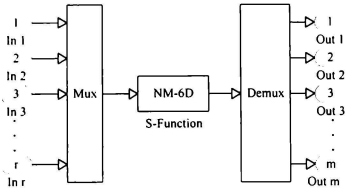


Figure 3. Connection of `NM-6D` for Linear Models

Executing the above command yields a linear aircraft model in a state-space representation,

$$\Delta\dot{\mathbf{X}} = \mathbf{A}\Delta\mathbf{X} + \mathbf{B}\Delta\mathbf{U}'$$

$$\Delta\mathbf{Y} = \mathbf{C}\Delta\mathbf{X} + \mathbf{D}\Delta\mathbf{U}'$$

where $\Delta\mathbf{X}$ and $\Delta\mathbf{U}'$ represents perturbed state vector, and input vector augmented with and disturbance vector, respectively. I.e.,

$$\Delta\mathbf{X} = [u, v, w, p, q, r, \phi, \theta, \varphi, x, y, z]^T$$

$$\Delta\mathbf{U}' = \begin{bmatrix} \Delta\mathbf{U} \\ \Delta\mathbf{W} \end{bmatrix}$$

As seen in the perturbed state vector $\Delta\mathbf{X}$, this model includes both longitudinal and lateral motion variables but it can be separated into longitudinal and lateral model of traditional flight dynamic analysis and control system design. Obtaining a longitudinal or a lateral model in this scheme provides improved model accuracy than those computed by analytical formulas based on stability derivatives, as given in standard textbooks such as reference 1 and 4.

Linear Longitudinal Models

Arranging perturbed state vector $\Delta\mathbf{X}$ as

$$\Delta\mathbf{X} = \begin{bmatrix} \Delta\mathbf{X}_{Lon} \\ \Delta\mathbf{X}_{Lat} \end{bmatrix}$$

then the longitudinal model can be obtained by

$$\Delta\dot{\mathbf{X}}_{Lon} = \mathbf{A}_{Lon}\Delta\mathbf{X}_{Lon} + \mathbf{B}_{Lon}\Delta\mathbf{U}_{Lon}$$

$$\Delta\mathbf{Y}_{Lon} = \mathbf{C}_{Lon}\Delta\mathbf{X}_{Lon} + \mathbf{D}_{Lon}\Delta\mathbf{U}_{Lon}$$

where

$$\Delta\mathbf{X}_{Lon} = [u, w, q, \theta, z]^T$$

$$\Delta\mathbf{U}_{Lon} = [\Delta N_e, \Delta\delta_e, \Delta\delta_M]^T$$

$$\Delta\mathbf{Y}_{Lon} = [\Delta V_E, \Delta\alpha, q, \theta, \Delta h, \Delta\gamma, \Delta n_x, \Delta n_z]^T$$

Note that the output vector is selected such that the model is suitable for flight control law synthesis. It is convenient to use $\Delta V_E, \Delta\alpha$ than u, w, so that the state vector is redefined as

$$\Delta\mathbf{X}_{Lon} = [\Delta V_E, \Delta\alpha, q, \theta, z]^T$$

with a little algebraic manipulations. Now, this longitudinal state-space model represents the SIMULINK® block library model `LMPss-4th`, as shown in Figure 4.

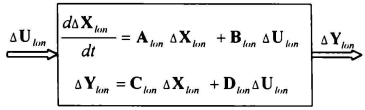


Figure 4. `LMPss-4th` Library Model

Occasionally, a transfer function model is required, it can be easily obtained using the `ss2tf` function command in MATLAB®, and this transfer function model constitutes `LMPtf-Yi/Uj-4th` as shown in Figure 5. Those transfer functions are all possible combinations of input to output variables, i.e.

$$\frac{\Delta V_E}{\Delta N_e}, \cdots, \frac{\Delta n_z}{\Delta\delta_M}$$

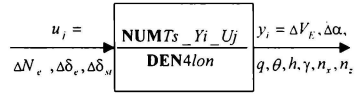


Figure 5. `LMPtf-Yi/Uj-4th` Library Model

Short-Period Approximation

Traditional approximation model for longitudinal modes, i.e. short-period and phugoid model can be obtained further from `LMPtf-Yi/Uj-4th`. For the short-period mode, the model is obtained by deleting corresponding low and column for the perturbed speed equation. Then,

$$\Delta\dot{\mathbf{X}}_{sp} = \mathbf{A}_{sp}\Delta\mathbf{X}_{sp} + \mathbf{B}_{sp}\Delta\mathbf{U}_{sp}$$

$$\Delta\mathbf{Y}_{sp} = \mathbf{C}_{sp}\Delta\mathbf{X}_{sp} + \mathbf{D}_{sp}\Delta\mathbf{U}_{sp}$$

where

$$\Delta \mathbf{X}_{sp} = [\Delta\alpha, q, \theta]^T$$

$$\Delta \mathbf{U}_{sp} = [\Delta\delta_e, \Delta\delta_M]^T$$

$$\Delta \mathbf{Y}_{sp} = [\Delta\alpha, q, \theta, \Delta\gamma, \Delta n_z]^T$$

$$\mathbf{A}_{sp} = \begin{bmatrix} \mathbf{A}_{Lon}(2,2) & 1 & 0 \\ \mathbf{A}_{Lon}(3,2) & \mathbf{A}_{Lon}(3,3) & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{B}_{sp} = \begin{bmatrix} \mathbf{B}_{Lon}(2,2) & \mathbf{B}_{Lon}(2,3) \\ \mathbf{B}_{Lon}(3,2) & \mathbf{B}_{Lon}(3,3) \\ 0 & 0 \end{bmatrix}, \mathbf{C}_{sp} = \begin{bmatrix} \text{eye}(3) \\ -1 \quad 0 \quad 1 \\ \mathbf{C}_{Lon}(5,2) \quad 0 \quad 0 \end{bmatrix}$$

$$\mathbf{D}_{sp} = \begin{bmatrix} \text{zeros}(4,2) \\ \mathbf{D}_{sp}(5,2) \quad \mathbf{D}_{sp}(5,3) \end{bmatrix}$$

This approximated state-space model represents the SIMULINK® block library model **LMPss-sp-2nd**, as shown in Figure 6.

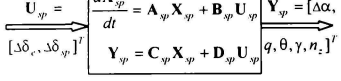


Figure 6. **LMPss-sp-2nd** Library Model

Similarly as in **LMPtf-Yi/Uj-4th** case, the transfer function model of short-period approximation is made into the SIMULINK® block library model **LMPtf-Yi/Uj-2nd_sp** as shown in Figure 7.

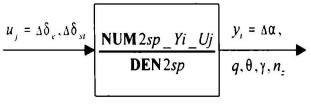


Figure 7..**LMPtf-Yi/Uj-2nd_sp** Library Model

Phugoid Approximation

Traditional approximation model for phugoid motion is obtained by setting $\Delta\dot{\alpha} = 0, \Delta\dot{q} = 0$. Then,

$$\begin{aligned} \Delta\dot{V}_E &= a_{11}\Delta V_E + a_{12}\Delta\alpha + \quad + a_{14}\theta + \\ &\quad + b_{11}\Delta N_e + b_{12}\Delta\delta_e + b_{13}\Delta\delta_M, \\ 0 &= a_{21}\Delta V_E + a_{22}\Delta\alpha + \quad q + a_{24}\theta + \\ &\quad + b_{21}\Delta N_e + b_{22}\Delta\delta_e + b_{23}\Delta\delta_M, \\ 0 &= a_{31}\Delta V_E + a_{32}\Delta\alpha + a_{33}q + \quad + \\ &\quad + b_{31}\Delta N_e + b_{32}\Delta\delta_e + b_{33}\Delta\delta_M, \\ \dot{\theta} &= q. \end{aligned}$$

After some matrix manipulations, these equations can be written as a state-space form given by

$$\Delta\dot{\mathbf{X}}_{ph} = \mathbf{A}_{ph}\Delta\mathbf{X}_{ph} + \mathbf{B}_{ph}\Delta\mathbf{U}_{ph}$$

$$\Delta\mathbf{Y}_{ph} = \mathbf{C}_{ph}\Delta\mathbf{X}_{ph} + \mathbf{D}_{ph}\Delta\mathbf{U}_{ph}$$

where

$$\Delta\mathbf{X}_{ph} = [\Delta V_E, \theta]^T, \quad \Delta\mathbf{U}_{ph} = [\Delta N_e, \Delta\delta_e, \Delta\delta_M]^T$$

$$\Delta\mathbf{Y}_{ph} = [\Delta\alpha, q, \Delta V_E, \theta]^T$$

$$\mathbf{A}_{ph} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12},$$

$$\mathbf{B}_{ph} = \mathbf{B}_2 - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{B}_1,$$

$$\mathbf{C}_{ph} = \begin{bmatrix} -\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \text{eye}(2) \end{bmatrix}, \mathbf{D}_{ph} = \begin{bmatrix} \mathbf{A}_{11}^{-1}\mathbf{B}_1 \\ \text{zeros}(2,3) \end{bmatrix}$$

$$\mathbf{A}_{11} = \begin{bmatrix} a_{22} & 1 \\ a_{21} & a_{23} \end{bmatrix}, \mathbf{A}_{12} = \begin{bmatrix} a_{21} & a_{24} \\ a_{31} & 0 \end{bmatrix}$$

$$\mathbf{A}_{21} = \begin{bmatrix} a_{12} & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{A}_{22} = \begin{bmatrix} a_{11} & a_{14} \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{B}_1 = \begin{bmatrix} b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}, \mathbf{B}_2 = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ 0 & 0 & 0 \end{bmatrix}$$

This approximated state-space model represents the SIMULINK® block library model **LMPss-ph-2nd**, as shown in Figure 8.

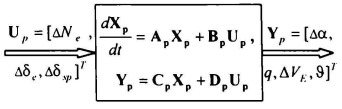


Figure 8. **LMPss-ph-2nd** Library Model

Similarly as in **LMPtf-Yi/Uj-4th** and **LMPtf-Yi/Uj-2nd_sp** cases, the transfer function model of phugoid approximation is made into the SIMULINK® block library model **LMPtf-Yi/Uj-2nd_ph** as shown in Figure 9.

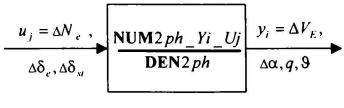


Figure 9. **LMPtf-Yi/Uj-2nd_ph** Model

Alternative Methods for Approximate Models

The approximate models discussed above are obtained by traditional, standard methods in classical textbooks such as reference 1. An alternative method to approximate these models is proposed in this study. The proposed method is based on applying partial fraction expansion using a prescribed MATLAB® command **residue**. This method results in more

accurate phugoid model than that by the classical approximation, although not so distinctive improvement is made in the short-period model. The proposed method starts with the models created by **LMPtf-Yi/Uj-4th**. Consider the next transfer function,

$$G^{y_i/\Delta u_j}(s) = \frac{y_i(s)}{\Delta\delta_e(s)}, \quad y_i = \Delta V_E, \Delta\alpha, \Delta q, \Delta\theta$$

These are the fourth order transfer functions, i.e

$$G^{y_i/\Delta u_j}(s) = \frac{n_{i_1}s^3 + n_{i_2}s^2 + n_{i_3}s + n_{i_4}}{s^4 + d_1s^3 + d_2s^2 + d_3s + d_4}.$$

Now, applying partial fraction expansion to this transfer function (using MATLAB® command **residue** command) leads to

$$G^{y_i/\Delta\delta_e}(s) = \frac{K_i^{sp}\left(T_{i_1}s+1\right)}{s^2 + 2\xi_{sp}\omega_{sp}s + \omega_{sp}^2} + \frac{K_i^{ph}\left(T_{i_2}s+1\right)}{s^2 + 2\xi_{ph}\omega_{ph}s + \omega_{ph}^2}$$

where $\xi_{sp}, \omega_{sp}, \xi_{ph}, \omega_{ph}$ are damping coefficient and natural frequency of short-period and phugoid motion. Since the system is linear, the output response $y_i(t)$ can be regarded as the sum of short-period and phugoid motion by the superposition principle, i.e.

$$y_i(t) = y_i^{sp}(t) + y_i^{ph}(t)$$

Furthermore, it can be written as

$$\frac{y_i^{sp}(s)}{\Delta\delta_e(s)} = \frac{K_i^{sp}\left(T_{i_1}s+1\right)}{s^2 + 2\xi_{sp}\omega_{sp}s + \omega_{sp}^2} \equiv W_i^{sp}(s)$$

$$\frac{y_i^{ph}(s)}{\Delta\delta_e(s)} = \frac{K_i^{ph}\left(T_{i_2}s+1\right)}{s^2 + 2\xi_{ph}\omega_{ph}s + \omega_{ph}^2} \equiv W_i^{ph}(s)$$

In case of a stable short-period as the most civil transport aircraft, its transient response quickly fades out and the further aircraft motion is the sum of steady-state short-period and phugoid motion as given by

$$y_i(t) = y_i^{sp}\big|_{ss} + y_i^{ph}(t)$$

This output response well represents the phugoid motion, and the approximated transfer function can be obtained by

$$G^{y_i/\Delta u_j}(s) = \frac{K_i^{sp}}{\omega_{sp}^2} + \frac{K_i^{ph}\left(T_{i_2}s+1\right)}{s^2 + 2\xi_{ph}\omega_{ph}s + \omega_{ph}^2}$$

$$= \frac{\frac{K_i^{sp}}{\omega_{sp}^2}s^2 + \left(\frac{2\xi_{ph}\omega_{ph}K_i^{ph}}{\omega_{sp}^2} + K_i^{ph}T_{i_1}\omega_{ph}^2\right)s + \left(\frac{K_i^{sp}\omega_{ph}^2}{\omega_{sp}^2} + K_i^{ph}\right)}{s^2 + 2\xi_{ph}\omega_{ph}s + \omega_{ph}^2}$$

This transfer function can be used for **LMPtf-Yi/Uj-2nd_ph** in the previous section. In addition, the short-period transfer function $W_i^{sp}(s)$ can also be used for **LMPtf-Yi/Uj-2nd_sp**. Next, the state-space form of corresponding transfer function models can replace **LMPss-sp-2nd** and **LMPss-ph-2nd**, accordingly, by using the MATLAB® command **tf2ss**. Comparisons are made in Figure 10.

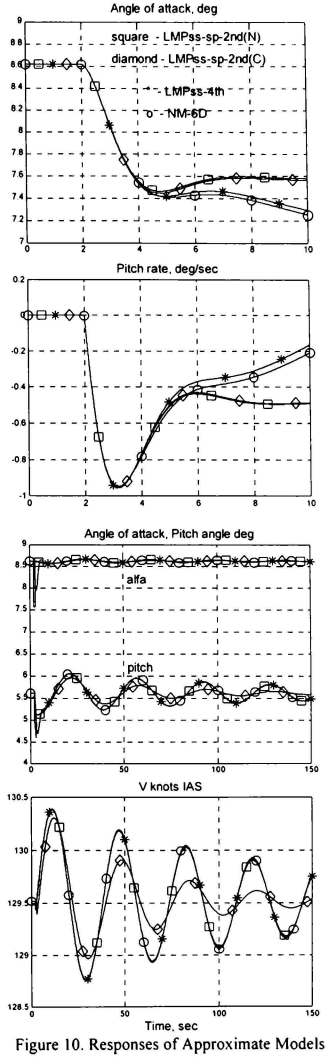


Figure 10. Responses of Approximate Models

The subject aircraft in this example is a regional twinjet transport during approach for landing. As expected, the approximated model obtained by the alternative method gives much accurate phugoid response.

Linear Lateral Models

A linear lateral model is obtained similarly as the longitudinal models in the previous section. The fourth order state-space model is

$$\Delta\dot{\mathbf{X}}_{\mathbf{Lat}} = \mathbf{A}_{\mathbf{Lat}}\Delta\mathbf{X}_{\mathbf{Lat}} + \mathbf{B}_{\mathbf{Lat}}\Delta\mathbf{U}_{\mathbf{Lat}}$$
$$\Delta\mathbf{Y}_{\mathbf{Lat}} = \mathbf{C}_{\mathbf{Lat}}\Delta\mathbf{X}_{\mathbf{Lat}} + \mathbf{D}_{\mathbf{Lat}}\Delta\mathbf{U}_{\mathbf{Lat}}$$

where

$$\Delta\mathbf{X}_{\mathbf{Lat}} = \left[\beta, p, r, \varphi\right]^T$$
$$\Delta\mathbf{U}_{\mathbf{Lat}} = \left[\Delta\delta_r, \Delta\delta_a\right]^T$$
$$\Delta\mathbf{Y}_{\mathbf{Lat}} = \left[\beta, p, r, \varphi, \Delta n_y\right]^T$$

and this represents the SIMULINK® block library model **`LMLss-4th`** and by converting into transfer function models, using the MATLAB® command **`ss2tf`** leads to the SIMULINK® block library model **`LMLtf-Yi/Uj-4th`**.

Dutch Roll Approximation

A traditional approximated model of Dutch-Roll motion is a coupled side-slip/yaw motion given by

$$\dot{\beta} = a_{11}\beta + a_{13}r + b_{11}\delta_r$$
$$\dot{r} = a_{31}\beta + a_{33}r + b_{31}\delta_r$$
$$\Delta n_y = c_{51}\beta + d_{51}\delta_r$$

This model usually exhibits significant discrepancies from the original fourth order model. In case partial fraction expansion method is applied to the original fourth order model, then much accurate approximated model can be obtained. Consider next transfer function models,

$$G^{y_i/\Delta\delta_r}(s) = \frac{n_{r_1}^{y_i}s^4 + n_{r_2}^{y_i}s^3 + n_{r_3}^{y_i}s^2 + n_{r_4}^{y_i}s + n_{r_5}^{y_i}}{s^4 + d_2s^3 + d_3s^2 + d_4s + d_5}$$

Applying partial fraction expansion to this transfer function gives

$$G^{y_i/\Delta\delta_r}(s) = \frac{k_{r_1}^{y_i}s + k_{r_2}^{y_i}}{s^2 + 2\xi_d\omega_d s + \omega_d^2} + \frac{k_{r_3}^{y_i}}{s - s_{roll}} + \frac{k_{r_4}^{y_i}}{s - s_{spiral}} + K_r^{y_i}$$

For a typical civil transport aircraft, the values of $k_{r_1}^{\beta}, k_{r_4}^{\beta}, k_{r_1}^{r}, k_{r_4}^{r}, k_{r_1}^{\Delta n_y}, k_{r_4}^{\Delta n_y}$ are usually negligible (i.e., insensitive to rudder input) and $K_r^{\beta} = K_r^{r} = 0$ and $K_r^{\Delta n_y} \neq 0$. This constitutes the transfer function models for lateral motion and the set of SIMULINK® block libraries, **`LMLtf-Yi/Uj-2nd`**. Further, the second order state-space models, **`LMLss-2nd`**, can be obtained by applying the MATLAB® command **`tf2ss`**. Figure 11 illustrates the comparison of lateral response of the same previous aircraft.

Roll Convergence Approximation

A typical model for the roll convergence approximation is given by

$$\Delta\dot{p} = a_{22}\Delta p + b_{22}\Delta\delta_a$$

but this model has a substantial difference when compared to the full 6 DOF nonlinear model. Again, an improved model can be obtained by applying the partial fraction expansion method to the following transfer function,

$$G^{p/\Delta\delta_a}(s) = \frac{k_{a_1}^{p}s + k_{a_2}^{p}}{s^2 + 2\xi_d\omega_d s + \omega_d^2} + \frac{k_{a_3}^{p}}{s - s_{roll}} + \frac{k_{a_4}^{p}}{s - s_{spiral}}$$

and by deleting dutch-roll and spiral components, the roll-convergence transfer function model, **`LMLtf-1st`**, is obtained as below.

$$G^{p/\Delta\delta_a}(s) = \frac{k_{a_3}^{p}}{s - s_{roll}}$$

Figure 12 shows the bank angle and roll-rate response of different approximate models to the aileron input.

Third order Approximate Models

Although seldom practically used, the third order approximate models can be similarly obtained using the partial fraction expansion method. They are obtained by deleting the component corresponding to spiral component. The third order transfer function models to rudder input are given by

$$G^{y_i/\Delta\delta_r}(s) = \frac{k_{r_1}^{y_i}s + k_{r_2}^{y_i}}{s^2 + 2\xi_d\omega_d s + \omega_d^2} + \frac{k_{r_3}^{y_i}}{s - s_{roll}} + K_r^{y_i}$$
$$= \frac{n_{r_1}^{y_i}s^3 + n_{r_2}^{y_i}s^2 + n_{r_3}^{y_i}s + n_{r_4}^{y_i}}{s^3 + d_2s^2 + d_3s + d_4}$$

where $y_i = \beta, p, r, \Delta n_y$ and

$$G^{\phi/\Delta\delta_r}(s) = G^{p/\Delta\delta_r}(s) + a_{43}G^{r/\Delta\delta_r}(s)$$

and to aileron input is are given by

$$G^{p/\Delta\delta_a}(s) = \frac{k_{a_1}^{p}s + k_{a_2}^{p}}{s^2 + 2\xi_d\omega_d s + \omega_d^2} + a_{43}\frac{k_{a_3}^{p}}{s - s_{roll}}$$

$$G^{\phi/\Delta\delta_a}(s) = G^{p/\Delta\delta_a}(s) + a_{43}G^{r/\Delta\delta_a}(s)$$

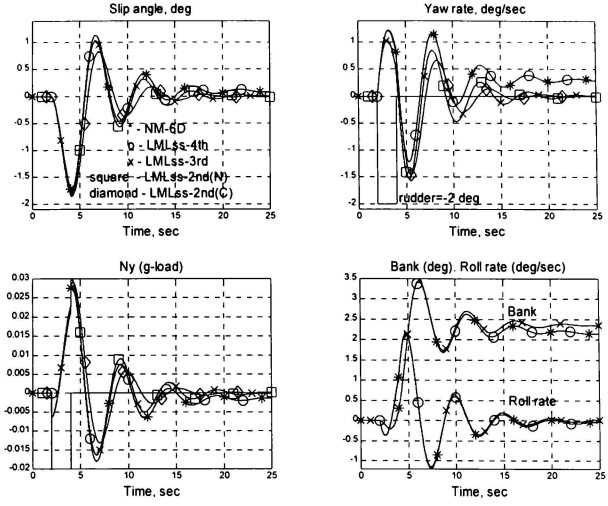


Figure 11 .Response of Lateral Approximate Models (Rudder Input)

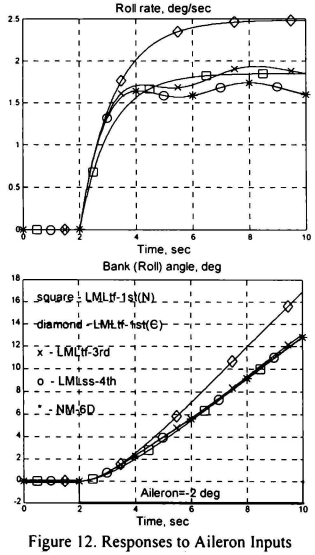


Figure 12. Responses to Aileron Inputs

PRACTICAL EQUILIBRIUM FLIGHT

Simulation of the full 6 DOF nonlinear flight equation requires an initial condition to proceed the integration of differential equations. The main parameters to determine the initial conditions are

- Weight configurations such as Operational Empty Weight (OEM, mg_o), Fuel Weight (mg_f), freight weight (mg_c), etc.
- Airplane configuration such as flaps & slats positions, landing gear position, etc.
- C.G. position
- Flight altitude
- Airspeed (V_{EAS}, V_{TAS}, M)
- Flight trajectory which is given by two angles, flight path angle ($\gamma = \gamma^*$) and heading angle (Ψ_v). The flight path angle can be replaced by engine condition ($N_{\infty} = N_{\infty}^*$)

The initial conditions are obtained from steady-state flight regimes, of which particularly interested in practical operations are given by:

1. Steady motion in the vertical plane at
   - constant flight speed ($V = V_{const}$)
   - constant, symmetrical thrust ($N_{\infty} = N_{\infty}^*$)
   - zero bank & side-slip angles ($\Phi = \beta = 0$)

Examples: Take-off, Steady climb, Climbing ascent, On-route climb, Climb on track, Descent, Idle descent, Emergency descent, Drift down, etc.

2. Steady motion in the vertical plane at
   - constant flight speed ($V = V_{const}$)
   - constant flight path angle ($\gamma = \gamma^{\circ}$)
   - constant, symmetrical thrust ($N_{ec} = N_{ec}^{\circ}$)
   - zero bank & side-slip angles ($\Phi = \beta = 0$)

Examples: Cruise at constant altitude, On-route flight, Box-pattern flight, Turning flight, Approach, etc.

3. Steady, straight motion at
   - constant flight speed ($V = V_{const}$)
   - constant thrust ($N_{ec} = N_{ec}^{\circ}$) of symmetrical and asymmetrical engine operation
   - constant flight speed ($V = V_{const}$)
   - given bank angle ($\Phi = \Phi^{\circ} = const$)
   - zero side-slip angle ($\beta = 0$)

Examples: Take-off, Steady climb, Climbing ascent, On-route climb, Climb on track, Descent, Idle descent, Emergency descent, Drift down, etc. with wind or with one or more engine failures

4. Steady, straight motion at
   - constant flight speed ($V = V_{const}$)
   - constant thrust ($N_{ec} = N_{ec}^{\circ}$) of symmetrical and asymmetrical engine operation
   - constant side-slip angle ($\beta = \beta^{\circ} = const$)
   - zero bank angle ($\Phi = 0$)

Examples: Take-off, Steady climb, Climbing ascent, On-route climb, Climb on track, Descent, Idle descent, Emergency descent, Drift down, etc. at flight with wind or with one or more engine failures.

5. Steady, straight motion at
   - constant flight speed ($V = V_{const}$)
   - flight path angle ($\gamma = \gamma^{\circ}$)
   - constant thrust ($N_{ec} = N_{ec}^{\circ}$) of symmetrical and asymmetrical engine operation
   - given bank angle ($\Phi = \Phi^{\circ} = const$)
   - zero side-slip angle ($\beta = 0$)

Examples: Cruise on constant altitude, On-route flight, Box-pattern flight, Turning flight, Approach, etc. at flight with wind or with one or more engine failures.

6. Steady, straight motion at
   - constant flight speed ($V = V_{const}$)
   - flight path angle ($\gamma = \gamma^{\circ}$)
   - constant thrust ($N_{ec} = N_{ec}^{\circ}$) of symmetrical and asymmetrical engine operation
   - constant side-slip angle ($\beta = \beta^{\circ} = const$)
   - zero bank angle ($\Phi = 0$)

Examples: Cruise on constant altitude, On-route flight, Box-pattern flight, Turning flight, Approach, etc. at flight with wind or with one or more engine failures.

7. A coordinate turn at
   - constant flight speed ($V = V_{const}$)
   - flight path angle ($\gamma = \gamma^{\circ}$)
   - constant thrust ($N_{ec} = N_{ec}^{\circ}$)
   - given bank angle ($\Phi = \Phi^{\circ} = const$)
   - zero side-slip angle ($\beta = 0$)

8. Take-off start on runway. Computation of the airplane c.g. position relative to the runway surface, pitch angle, undercarriage and tire deflections

Trim (Steady-State Flight) Solutions

Finding necessary initial condition requires solving the systems of nonlinear equations by setting ${}^{E}\dot{\vec{V}}_{B}^{B^{\circ}} = {}^{N}\dot{\vec{\omega}}_{B}^{B} = 0$ of the equations given in the introduction. These equations should be solved by iterative procedure [5], and the procedure applied in the present simulation software, which is combination of step-by-step false position method and Newton iterative method, is described as follow:

Step 1. Starting point – set data for a flight mode

Main: $mg = mg_a + mg_f + mg_c$

$\tilde{\vec{R}}_{B}^{B;B^{\circ}}, \delta_{flap}, \delta_{slat}, \delta_{LG}, h_a, V_{IAS}, N_{ec}$ (or γ°)

Additional: $\Phi^{\circ}(\beta^{\circ}), \Psi^{\circ}, X^{\circ}, Y^{\circ}$, etc.

Step 2. Compute ISA parameters.

$\rho,\ \sigma = \sqrt{\dfrac{\rho(h = h^{\circ})}{\rho_0(h = 0)}},\ g,\ T(^{\circ}\mathbf{K}), p = f(h^{\circ})$

and compute $V_a, V_k, M, T' = f\left(N_{ec}^{\circ}, M^{\circ}, h^{\circ}, T\right)$ at an engine condition, if given.

Step 3. Selecting initial parameters for

$N_{ec_i}^{1} = 0.5(N_{ec_i}^{idle} + N_{ec_i}^{takeoff})$ in case γ° is given.

$\Theta = \Theta^{1}$, in case $N_{ec_i}^{\circ}$ is given.

$\alpha = \alpha^{1}, \beta = \beta^{1}$ in case Φ° is given.

$\Phi = \Phi^{1}$ in case β° is given.

$\delta_a^{1}, \delta_{f,s}^{1} = f\left(\delta_a^{1}\right), \delta_u^{1}(\delta_r^{1}), \delta_r^{1}$

Step 4. Main Loop

- Step 4.1: Apply false position method to find an angle of attack α^{k+1} from the Z-body axis force equation $Z(\alpha^{k+1})=0$ while fixing other parameters, $N_{\alpha_c}^k$ (or Θ^k), δ_u^k (or δ_e^k), δ_r^k, β^k (or Φ^k), δ_a^k, $\delta_{fs}^k = f(\delta_a^k)$.
- Step 4.2: Apply false position method to find δ_u^{k+1} (or δ_e^{k+1}) from the Y-body axis moment equation $M(\delta_u^{k+1} \text{ or } \delta_e^{k+1})=0$ while fixing other parameters, $\alpha^{k+1}, N_{\alpha_c}^k$ (or Φ^k), δ_r^k, β^k (or Φ^k), δ_a^k, $\delta_{fs}^k = f(\delta_a^k)$.
- Step 4.3: Find Θ^{k+1}, at a given engine condition from the X-body axis force equation $X(\Theta_{k+1})=0$ while fixing the parameters α_{k+1}, δ_u^{k+1} (or δ_e^{k+1}), δ_r^k, β^k (or Φ^k), δ_a^k, $\delta_{fs}^k = f(\delta_a^k)$. In case a flight path angle γ° is given, the X-body force equation is used to find an engine condition from $F^1(N_{\alpha_c}^{k+1})=0$ by applying false position method. For this case, it is necessary to find Θ^{k+1} from

$$\sin\gamma = (\sin\Theta\cos\alpha - \cos\Theta\sin\alpha\cos\Phi)\cos\beta - \cos\Theta\sin\Phi\sin\beta$$

before executing the main loop. The Newton iterative method is used to find

$$\Theta^{k+1} = \Theta^k - \frac{f(\Theta^k)}{f'(\Theta^k)}$$

where

$$f(\Theta) = \sin\gamma - (\sin\Theta\cos\alpha - \cos\Theta\sin\alpha\cos\Phi)\cos\beta + \cos\Theta\sin\Phi\sin\beta$$

- Step 4.4: Apply false position method to find β_{k+1} at a given Φ° from the Y-body axis force equation $Y(\beta_{k+1})=0$ while fixing other parameters, α^{k+1}, δ_u^{k+1} (or δ_e^{k+1}), $N_{\alpha_c}^{k+1}$ (or Φ^{k+1}), δ_r^k, δ_a^k, $\delta_{fs}^k = f(\delta_a^k)$.
- Step 4.5 & 4.6: Apply false position method to find δ_a^{k+1} and δ_r^{k+1} from X and Z body axes moment equation $\mathbf{L}(\delta_a^{k+1}, \delta_{fs}^{k+1}), N(\delta_r^{k+1})$ while fixing other parameters α^{k+1}, δ_u^{k+1} (or δ_e^{k+1}), $N_{\alpha_c}^{k+1}$ (or Φ^{k+1}), β^{k+1} (or Φ^{k+1}), δ_r^k.
- Step 4.7: Check the conditions

$$X, Y, Z, \mathbf{L}, M, N \le eps$$

is satisfied, and escape from the loop and proceed to the next step. Otherwise, continue looping.

Step 5. Setting up an initial condition vector

$$\mathbf{X}^\circ = [V_t^\circ\cos\beta^\circ\cos\alpha^\circ,\ V_t^\circ\sin\beta^\circ,\ V_t^\circ\cos\beta^\circ\sin\alpha^\circ,\ 0,\ 0,\ 0,\ \Phi^\circ\ \Theta^\circ\ \Psi^\circ,\ x^\circ,\ y^\circ,\ z^\circ]$$

$$\mathbf{U}^\circ = [N_{\alpha_c}^\circ, \delta_{flap}^\circ, \delta_{vlat}^\circ, \delta_{TG}^\circ, \delta_e^\circ, \delta_a^\circ, \delta_{fs}^\circ, \delta_r^\circ]^t$$

$$\mathbf{Y}^\circ = g(\mathbf{X}^\circ, \mathbf{U}^\circ)$$

SIMULATION PROCEDURES, RESULTS ANALYSIS & USER INTERFACE

Figure 13 shows the overall procedure of the current software operation. At the first end, the flight mode of interest is specified. Then, the values of initial condition to the corresponding initial flight mode are computed for all models in Table 1. With the graphical user interface shown in Figure 14, the user can easily perform necessary analysis and synthesis required for the aircraft development including

- Computation of system parameters for the initial flight model such as the angle of attack, flight path angle at a given power setting (or to compute engine conditions if flight path angle is given), and trim positions of control surfaces and control levers.
- Computation of system parameters for all linearized models of the airplane motion.
- Computation of stability characteristics such as damping coefficients and natural frequencies, time constants, transient response, etc.
- Computation of control characteristics such as the controllability indices at speed, g-load, rate of roll, etc.
- Comparision of linear and non-linear models of aircrat motion.
- Choice of an input control or disturbance action and the visualization and the storage for the post process of simulation.

SOFTWARE APPLICATION

With the developed graphical SIMULINK® library models, control law synthesis for prototype aircraft can be studied, compared and verified at the tip of control system designer's hand. In addition, various control design methodologies ranging from a classical method to a modern robust control method, can be applied without much difficult by using MATLAB® toolboxes such as 'Control System Toolbox', 'Robust Control Toolbox', 'μ-Synthesis Toolbox', etc. Pros and cons among different types of control methodologies can be easily understood. Furthermore, the full 6 DOF nonlinear simulation shows controller performance with respect to nonlinearities and uncertainties, and provides the baseline for controller gain scheduling.

CONCLUSION

The set of graphical models of an airplane motion developed in this research allows constructing complex models of an airplane and its system for a flight condition of interest. These models can be use for the development of flight control system as well as to be used for the engineering simulator source code.

REFERENCE

[1] Etkin, B. "Dynamics of Atmospheric Flight," John Wiley & Sons, Inc. 1972.

[2] Using SIMULINK®, Mathwork Inc.

[3] Verspay, J. J. L. H.; de Myunck, R. J.; Nibbelke, R. J.; van den Bosch, J. J. , "Simulator evaluation on control and display issues for a future regional aircraft" AIAA Paper 96-3382, AIAA, Atmospheric Flight Mechanics Conference, San Diego, CA, July 29-31, 1996.

[4] Steven, B.L. & Lewis F.L., "Aircraft Control and Simulation," John Wiley & Sons, Inc. 1992.

[5] "Numerical Recipes in C. The art of scientific computing", William H. Press, and others. Cambridge University Press, 199

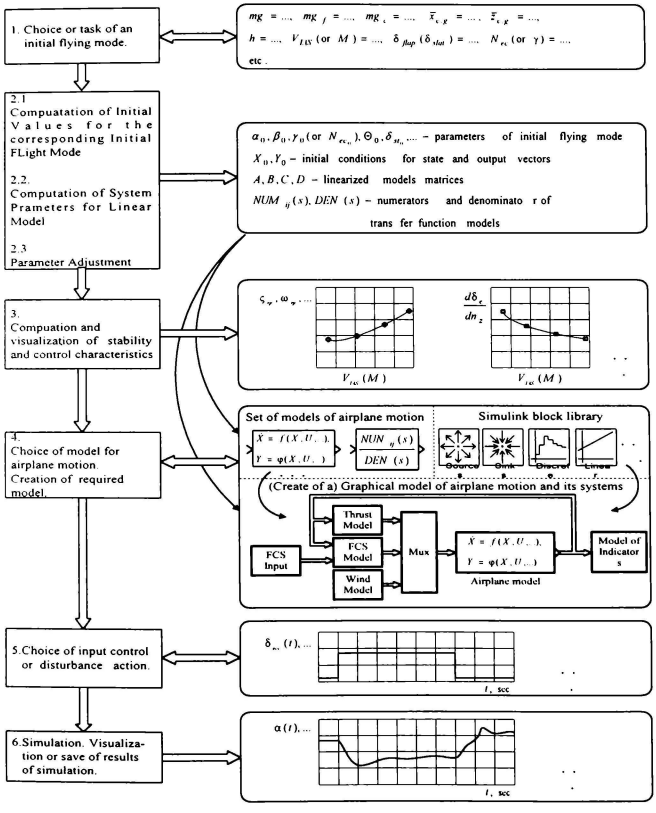


Figure 13. Software Operation Procedure

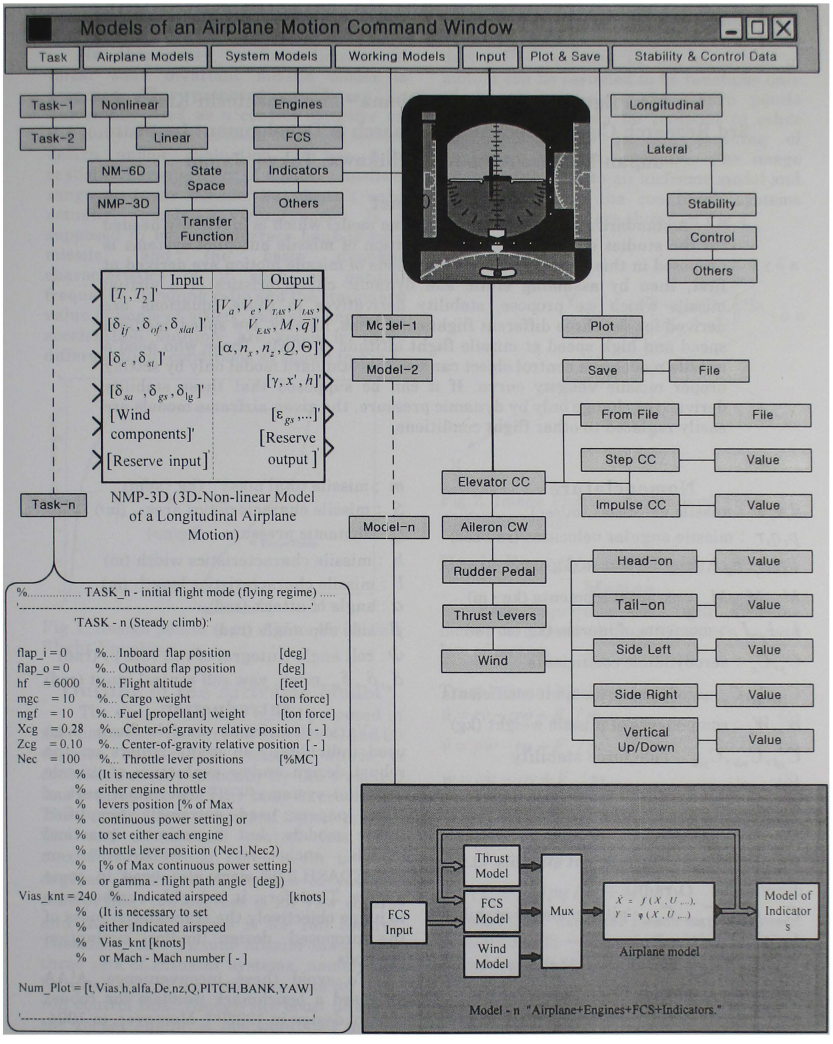


Figure 14. Graphical User Interface

Standard Airframe Model for Missile Autopilot Design and Evaluation

Hirofumi Eguchi*1, Kazumitu Obana*2 and Masatoshi Kiso*2
3rd Research Center, Technical Research & Development Institute,
Japan Defense Agency, Tachikawa, Tokyo, Japan

Abstract

A standard air-to-air missile airframe model which is inevitably needed for the studies on design and/or evaluation of missile autopilot systems is proposed in this paper. The basic equations of missile motion are derived at first, then by assuming static and dynamic characteristics of a virtual missile which we propose, stability derivatives in those equations are derived for the three different flight conditions, namely, low speed, medium speed and high speed at missile flight altitude of 30Kft. Those who need a missile model as a control object can use this standard model only by setting proper missile velocity curve. If it can be supposed that these stability derivatives change only by dynamic pressure, the given airframe models are easily replaced to other flight conditions.

Nomenclature

u, v, w : missile velocities (m/sec)

p, q, r : missile angular velocities (rad/sec)

F_x, F_y, F_z : external forces (kg)

M_x, M_y, M_z : external moments (kg · m)

I_x, I_y, I_z : moments of inertia (kg · m · s²)

C_y, C_z : aerodynamic coefficients

C_l, C_m, C_n :aerodynamic moment coefficients

W_y, W_z : components of missile weight (kg)

$C_{y\beta}, C_{y\delta r}, C_{y\delta a}$: side force stability derivatives

$C_{z\alpha}, C_{z\delta e}, C_{z\delta r}, C_{z\delta a}$: longitudinal force stability derivatives

$C_{n\beta}, C_{n\delta r}, C_{n\delta a}$: yaw moment stability derivatives (1/rad)

$C_{m\alpha}, C_{m\delta e}, C_{m\delta r}, C_{m\delta a}$: pitch moment stability derivatives (1/rad)

$C_{l\beta}, C_{l\delta r}, C_{l\delta a}$: roll moment stability derivatives (1/rad)

m : missile total mass (kg · s²/m)

S : missile characteristics area (m²)

Q : dynamic pressure (kg/m²)

b : missile characteristics width (m)

l : missile characteristics length (m)

α : angle of attack (rad)

β : side slip angle (rad)

Φ : roll angle (Integral of roll rate p) (rad)

$\delta_e, \delta_r, \delta_a$: pitch, yaw,roll control input (rad)

Introduction

Various airframe models have been used individually in many papers on the robust design and/or evaluation of missile autopilot systems.[1]~[17] The major part of these papers used extremely simplified linear models, but sometimes, practical models such as EMRAAT[3] or HAVEDASH II [13] were also used in some papers. Therefore, it is highly cumbersome to judge objectively the true effectiveness of the proposed design and/or evaluation methods.

To avoid these inconvenience, AIAA proposed a benchmark problem for robust control design in 1992[10]. Moreover, in 1994, control design challenge was planned for an given aircraft model[12]. The aircraft model proposed by AIAA in this time was

*1:System Engineer
*2:Research Engineer
Copyright c 1999 The American Institute of Aeronautics and Astronautics Inc, All right reserved.

considerably practical one, and it included many nonlinear elements in actuator and engine models. Perhaps for this nonlinear elements, not so many control system designers could apply to this control design challenge.

In this paper, a finite-dimensional linear time invariant missile model is proposed. This proposed model can be universally used as a control objective on the study of robust missile autopilot system design and/or evaluation. It has fully realistic characteristics of a short / medium range air-to-air missile, but doesn't exist actually, namely, a virtual model. It is supposed to be a 90 degrees bank-to-turn missile, and the basic dynamic characteristics of it such as natural frequency, damping ratio and the stationary value of angle of attack, side slip angle and acceleration multiple are given in three different flight conditions as shown in Fig.1.

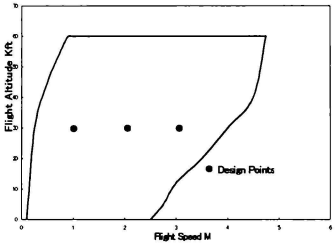


Fig.1. Design points in an Air-to-Air Missile Flight Envelope

Outline of the Airframe Model

The virtual airframe model proposed in this paper is symmetry up and down, and 90 degrees bunk-to-turn missile which is controlled aerodynamically by the 4 control fins attached to the rear body in cruciform. The 4 control fins consists of 2 pitch control fins and 2 yaw-roll control fins. The maximum deflection of each fin is ± 10 degrees. In the ± 10 degrees of yaw-roll control fins, ± 5 degrees is for yaw control and the rest ± 5dgrees is for roll control. Therefore, this virtual missile must have three sets of actuator systems, namely one for pitch control fins and two sets for yaw-roll control fins. Though the boost phase of the rocket engine is not simulated in this model, missile velocity is treated as a parameter at 30Kft flight altitude. In actual missile development, aerodynamic stability derivatives are obtained from the wind tunnel test data, and the data for the velocity of 1 Mach can't be obtained. But in this paper, flight condition of 1 Mach is also selected for the sake of convenience as a standard airframe model. If these stability derivatives included in the equations of motion can be assumed to be functions only of dynamic pressure, the design points shown in Fig.1 can be replaced to other equivalent points in the meaning of equivalent dynamic pressure. The image picture of this air-to-air airframe model and the definition of the coordinate systems used in this paper are shown in Fig.2.

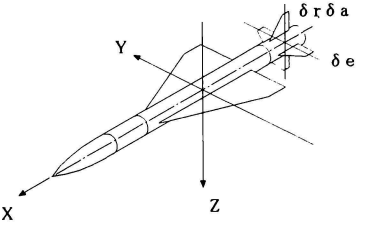


Fig.2. Image Picture of Standard Air-to-Air Airframe Model

Basic Equations of Rigid Airframe Motion

Basic equations for rigid airframe motion are derived from Newton's second law of motion.

Translational Motion

$$\dot{u} = rv - qw + F_x / m \tag{1}$$

$$\dot{v} = pw - ru + F_y / m \tag{2}$$

$$\dot{w} = qu - pv + F_z / m \tag{3}$$

Rotational Motion

$$\dot{p} = (I_y - I_z)qr / I_x + M_x / I_x \tag{4}$$

$$\dot{q} = (I_z - I_x)rp / I_y + M_y / I_y \tag{5}$$

$$\dot{r} = (I_x - I_y)pq / I_z + M_z / I_z \tag{6}$$

Assume that, external forces and moments in equation(2) ~ (6) can be linearized as equations (7)~(11).

$$\begin{aligned} F_y &= QS \cdot C_y(\beta, \delta_r, \delta_a) + W_y \\ &= QS(C_{y\beta}\beta + C_{y\delta_r}\delta_r + C_{y\delta a}\delta_a) \\ &\quad + mg \cdot Sin\Phi Cos\theta \end{aligned} \tag{7}$$

$$F_z = QS \cdot C_z(\alpha, \delta_e, \delta_r, \delta_a) + W_z$$
$$= QS(C_{z\alpha}\alpha + C_{z\delta e}\delta_e + C_{z\delta r}\delta_r + C_{z\delta a}\delta_a) + mg \cdot Cos\Phi Cos\theta \quad (8)$$

$$M_x = QSb \cdot C_l(\beta, \delta_r, \delta_a)$$
$$= QSb(C_{l\beta}\beta + C_{l\delta r}\delta_r + C_{l\delta a}\delta_a) \quad (9)$$

$$M_y = QSl \cdot C_m(\alpha, \delta_e, \delta_r, \delta_a)$$
$$= QSl(C_{m\alpha}\alpha + C_{m\delta e}\delta_e + C_{m\delta r}\delta_r + C_{m\delta a}\delta_a) \quad (10)$$

$$M_z = QSl \cdot C_n(\beta, \delta_r, \delta_a)$$
$$= QSl(C_{n\beta}\beta + C_{n\delta r}\delta_r + C_{n\delta a}\delta_a) \quad (11)$$

Moreover, the following approximations are applied.

$u = v_m$ (const.)

$I_y = I_z$, $I_x \ll I_y$, $I_x \ll I_z$

$v \fallingdotseq \beta V_m$ (12)

$w \fallingdotseq \alpha V_m$

$\Phi \ll 1$

Then rigid airframe equations of motion are obtained.

$$u = v_m \text{ (const.)} \quad (13)$$

$$\dot{\alpha} = q - \beta p + QS(C_{z\alpha}\alpha + C_{z\delta e}\delta_e + C_{z\delta r}\delta_r + C_{z\delta a}\delta_a)/mv_m + gCos\theta/v_m \quad (14)$$

$$\dot{\beta} = \alpha p - r + QS(C_{y\beta}\beta + C_{y\delta r}\delta_r + C_{y\delta a}\delta_a)/mv_m + g\Phi \cdot Cos\theta/v_m \quad (15)$$

$$\dot{p} = QSb(C_{l\beta}\beta + C_{l\delta r}\delta_r + C_{l\delta a}\delta_a)/I_x \quad (16)$$

$$\dot{q} = rp + QSl(C_{m\alpha}\alpha + C_{m\delta e}\delta_e + C_{m\delta r}\delta_r + C_{m\delta a}\delta_a)/I_y \quad (17)$$

$$\dot{r} = -pq + QSl(C_{n\beta}\beta + C_{n\delta r}\delta_r + C_{n\delta a}\delta_a)/I_z \quad (18)$$

In above equations, substitute as follows for convenience,

$$z_\alpha = QS \cdot C_{z\alpha}/mv_m$$
$$z_{\delta e} = QS \cdot C_{z\delta e}/mv_m$$
$$z_{\delta r} = QS \cdot C_{z\delta r}/mv_m$$
$$z_{\delta a} = QS \cdot C_{z\delta a}/mv_m$$
$$m_\alpha = QSl \cdot C_{m\alpha}/I_y$$
$$m_{\delta e} = QSl \cdot C_{m\delta e}/I_y$$
$$m_{\delta r} = QSl \cdot C_{m\delta r}/I_y$$
$$m_{\delta a} = QSl \cdot C_{m\delta a}/I_y$$
$$y_\beta = QS \cdot C_{y\beta}/mv_m \quad (19)$$
$$y_{\delta r} = QS \cdot C_{y\delta r}/mv_m$$
$$y_{\delta a} = QS \cdot C_{y\delta a}/mv_m$$
$$n_\beta = QSL \cdot C_{n\beta}/I_z$$
$$n_{\delta r} = QSl \cdot C_{n\delta r}/I_z$$
$$n_{\delta a} = QSl \cdot C_{n\delta a}/I_z$$
$$l_\beta = QSbC_{l\beta}/I_x$$
$$l_{\delta r} = QSb \cdot C_{l\delta r}/I_x$$
$$l_{\delta a} = QSbC_{l\delta a}/I_x$$

Then, the similar equations of rigid airframe motion that K.A.Wise[5] and C.F.Lin[13] used in their paper respectively are obtained.

$$\dot{\alpha} = z_\alpha\alpha + q - \beta p + z_{\delta e}\delta_e + z_{\delta r}\delta_r + z_{\delta a}\delta_a + gCos\theta/v_m \quad (20)$$

$$\dot{q} = m_\alpha\alpha + rp + m_{\delta e}\delta_e + m_{\delta r}\delta_r + m_{\delta a}\delta_a \quad (21)$$

$$\dot{\beta} = y_\beta\beta - r + \alpha p + y_{\delta r}\delta_r + y_{\delta a}\delta_a + g\Phi \cdot Cos\theta/v_m \quad (22)$$

$$\dot{r} = n_\beta\beta - pq + n_{\delta r}\delta_r + n_{\delta a}\delta_a \quad (23)$$

$$\dot{p} = l_\beta\beta + l_{\delta r}\delta_r + l_{\delta a}\delta_a \quad (24)$$

$$\dot{\Phi} = p \quad (25)$$

Here, we assume more boldly.

1 Pitch system can be absolutely separated from yaw and roll system.

2 Gyroscopic moment in yaw system can be neglected.

3 Coriolis force in yaw system can be treated for constant angle of attack.

4 $\theta_0 \ll 1$.

Then, we get the following equations.

$$\dot{\alpha} = z_\alpha\alpha + q + z_{\delta e}\delta_e \quad (26)$$

$$\dot{q} = m_\alpha\alpha + m_{\delta e}\delta_e \quad (27)$$

$$\dot{\beta} = y_\beta \beta - r + \alpha_0 p + g\Phi / v_m + y_{\delta r}\delta_r + y_{\delta a}\delta_a \quad (28)$$

$$\dot{r} = n_\beta \beta + n_{\delta r}\delta_r + n_{\delta a}\delta_a \quad (29)$$

$$\dot{p} = l_\beta \beta + l_{\delta r}\delta_r + l_{\delta a}\delta_a \quad (30)$$

$$\dot{\Phi} = p \quad (31)$$

In state space form,

$$\dot{X} = AX + BU \quad (32)$$

where,

$$X = \begin{bmatrix} \alpha \\ q \\ \beta \\ r \\ p \\ \Phi \end{bmatrix} \quad A = \begin{bmatrix} z_\alpha & 1 & 0 & 0 & 0 & 0 \\ m_\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_\beta & -1 & \alpha_0 & g/v_m \\ 0 & 0 & n_\beta & 0 & 0 & 0 \\ 0 & 0 & l_\beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (33)$$

$$B = \begin{bmatrix} z_{\delta e} & 0 & 0 \\ m_{\delta e} & 0 & 0 \\ 0 & y_{\delta r} & y_{\delta a} \\ 0 & n_{\delta r} & n_{\delta a} \\ 0 & l_{\delta r} & l_{\delta a} \\ 0 & 0 & 0 \end{bmatrix} \quad U = \begin{bmatrix} \delta_e \\ \delta_r \\ \delta_a \end{bmatrix} \quad (34)$$

Pitch Dynamics Characteristics Analysis

Pitch flight control equations of motion are

$$\dot{\alpha} = z_\alpha \alpha + q + z_{\delta e}\delta_e \quad (26)$$

$$\dot{q} = m_\alpha \alpha + m_{\delta e}\delta_e \quad (27)$$

In state space form,

$$\dot{X} = AX + Bu \quad (35)$$

$$X = \begin{bmatrix} \alpha \\ q \end{bmatrix} \quad A = \begin{bmatrix} z_\alpha & 1 \\ m_\alpha & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} z_{\delta e} \\ m_{\delta e} \end{bmatrix} \quad u = \delta_e \quad (36)$$

Here, X is given as

$$X = [sI - A]^{-1} Bu \quad (37)$$

The transfer function description of the pitch dynamics for control input δ e are

$$\frac{\alpha}{\delta_e} = \frac{z_{\delta e}s + m_{\delta e}}{s^2 - z_\alpha s - m_\alpha} \quad (38)$$

$$\frac{q}{\delta_e} = \frac{m_{\delta e}s + z_{\delta e}m_\alpha - m_{\delta e}z_\alpha}{s^2 - z_\alpha s - m_\alpha} \quad (39)$$

Therefore, the pitch dynamics stability conditions are

$$m_\alpha < 0 \quad (40)$$

$$z_\alpha < 0 \quad (41)$$

The natural frequency and damping ratio of the pitch dynamics are given by equation (42) and (43).

$$\omega_n = \sqrt{-m_\alpha} \quad (42)$$

$$\varsigma = \frac{-z_\alpha}{2\sqrt{-m_\alpha}} \quad (43)$$

The stationary value of angle of attack α_{sta} for the step control input δ e is given by equation (44).

$$\alpha_{sta} = \lim_{s\to 0} \frac{z_{\delta e}s + m_{\delta e}}{s^2 - z_\alpha s - m_\alpha} \cdot \frac{\delta_e}{s} \cdot s = \frac{m_{\delta e}}{-m_\alpha}\delta_e \quad (44)$$

The stationary acceleration multiple obtained by the step control input δ e is represented by equation (45).

$$n_{zsta} = \frac{V_m}{g}(\dot{\alpha} - q)_{sta} = \frac{V_m}{g}(z_\alpha \alpha_{sta} + z_{\delta e}\delta_e) = \frac{V_m}{g}\left(\frac{z_\alpha m_{\delta e}}{-m_\alpha} + z_{\delta e}\right)\delta_e \quad (45)$$

Here, from Table3, $Z_{\delta e} < 0$, the relation of equation (46) must be satisfied.

$$|\alpha_{sta}| > \frac{1}{2\varsigma\omega_n}\frac{|n_{zsta}g|}{V_m} \quad (46)$$

Yaw-Roll Dynamics Characteristics Analysis

Yaw-Roll flight control equations of motion are

$$\dot{\beta} = y_\beta \beta - r + \alpha_0 p + g\Phi / v_m + y_{\delta r}\delta_r + y_{\delta a}\delta_a \quad (28)$$

$$\dot{r} = n_\beta \beta + n_{\delta r}\delta_r + n_{\delta a}\delta_a \quad (29)$$

$$\dot{p} = l_\beta \beta + l_{\delta r}\delta_r + l_{\delta a}\delta_a \quad (30)$$

$$\dot{\Phi} = p \quad (31)$$

In state space form,

$$\dot{X} = AX + BU$$

$$A = \begin{bmatrix} y_\beta & -1 & \alpha_0 & g/v_m \\ n_\beta & 0 & 0 & 0 \\ l_\beta & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (47)$$

$$B = \begin{bmatrix} y_{\delta r} & y_{\delta a} \\ n_{\delta r} & n_{\delta a} \\ l_{\delta r} & l_{\delta a} \\ 0 & 0 \end{bmatrix} \quad U = \begin{bmatrix} \delta_r \\ \delta_a \end{bmatrix}$$

Here, X is given as

$$X = [SI - A]^{-1} BU$$

$$[sI - A]^{-1} = \frac{\begin{bmatrix} s^3 & -s^2 & \alpha_0 s^2 + gs/v_m & gs^2/v_m \\ n_\beta s^2 & s^2(s - y_\beta) - \alpha_0 l_\beta s - g l_\beta / v_m & n_\beta(\alpha_0 s + g/v_m) & g n_\beta s/v_m \\ l_\beta s^2 & -l_\beta s & s^2(s - y_\beta) + n_\beta s & g l_\beta s/v_m \\ l_\beta s & -l_\beta & s(s - y_\beta) + n_\beta & s^2(s - y_\beta) + s(n_\beta - \alpha_0 l_\beta) \end{bmatrix}}{s\{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m\}} \quad (48)$$

The transfer function description of the yaw-roll dynamics for yaw control input δ r and roll control input δ a are equation (49)-(52).

$$\frac{\beta}{\delta_r} = \frac{y_{\delta r}s^2 - (n_{\delta r} - \alpha_0 l_{\delta r})s + g l_{\delta r}/v_m}{\{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m\}} \quad (49)$$

$$\frac{\beta}{\delta_a} = \frac{y_{\delta a}s^2 - (n_{\delta a} - \alpha_0 l_{\delta a})s + g l_{\delta a}/v_m}{\{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m\}}$$

$$\frac{r}{\delta_r} = \frac{n_{\delta r}s^3 + (n_\beta y_{\delta r} - y_\beta n_{\delta r})s^2 + \alpha_0(l_{\delta r}n_\beta - n_{\delta r}l_\beta)s + g(l_{\delta r}n_\beta - n_{\delta r}l_\beta)/v_m}{s\{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m\}} \quad (50)$$

$$\frac{r}{\delta_a} = \frac{n_{\delta a}s^3 + (n_\beta y_{\delta a} - y_\beta n_{\delta a})s^2 + \alpha_0(l_{\delta a}n_\beta - n_{\delta a}l_\beta)s + g(l_{\delta a}n_\beta - n_{\delta a}l_\beta)/v_m}{s\{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m\}}$$

$$\frac{p}{\delta_r} = \frac{l_{\delta r}s^2 + (y_{\delta r}l_\beta - l_{\delta r}y_\beta)s + (l_{\delta r}n_\beta - n_{\delta r}l_\beta)}{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m} \quad (51)$$

$$\frac{p}{\delta_a} = \frac{l_{\delta a}s^2 + (y_{\delta a}l_\beta - l_{\delta a}y_\beta)s + (l_{\delta a}n_\beta - n_{\delta a}l_\beta)}{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m}$$

$$\frac{\Phi}{\delta_r} = \frac{l_{\delta r}s^2 + (y_{\delta r}l_\beta - l_{\delta r}y_\beta)s + (l_{\delta r}n_\beta - n_{\delta r}l_\beta)}{s\{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m\}} \quad (52)$$

$$\frac{\Phi}{\delta_a} = \frac{l_{\delta a}s^2 + (y_{\delta a}l_\beta - l_{\delta a}y_\beta)s + (l_{\delta a}n_\beta - n_{\delta a}l_\beta)}{s\{s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m\}}$$

The basic characteristics of yaw-roll system can be obtained by equation (53).

$$s^3 - y_\beta s^2 + (n_\beta - \alpha_0 l_\beta)s - g l_\beta / v_m = 0 \quad (53)$$

The stability conditions for yaw-roll dynamics are equation (54).

$$\begin{aligned} & y_\beta < 0 \\ & n_\beta - \alpha_0 l_\beta > 0 \\ & l_\beta < 0 \\ & l_\beta g / v_m - y_\beta(n_\beta - \alpha_0 l_\beta) > 0 \end{aligned} \quad (54)$$

Here, if we assume that the gravity term in equation (53) doesn't affect seriously in yaw and roll dynamics, then, natural frequency is

$$\omega_n = \sqrt{n_\beta - \alpha_0 l_\beta} \quad (55)$$

damping ratio is

$$\varsigma = \frac{-y_\beta}{2\sqrt{n_\beta - \alpha_0 l_\beta}} \quad (56)$$

the stationary value of side slip angle for the control input δ r is
when $\alpha_0 = 0$

$$\beta_{sta} = -\frac{n_{\delta r}}{n_\beta}\delta_r \qquad (57)$$

when $\alpha_0 \neq 0$

$$\beta_{sta} = -\frac{(n_{\delta r} - \alpha_0 l_{\delta r})}{n_\beta - \alpha_0 l_\beta}\delta_r \qquad (58)$$

and the stationary acceleration multiple for zero angle of attack obtained by the control input δ r is

$$n_{ysta} = \frac{V_m}{g}(\dot{\beta} + r)_{sta} = \frac{V_m}{g}(\frac{y_\beta n_{\delta r}}{-n_\beta} + y_{\delta r})\delta_r \qquad (59)$$

Here, from Table 3, $y_{\delta r} > 0$, the relation of equation (60) must be satisfied.

$$|\beta_{sta}| > \frac{1}{2\zeta\omega_n}\frac{|n_{ysta}g|}{V_m} \quad \text{for } \alpha_0 = 0 \qquad (60)$$

Also, from Table 3, $l_{\delta r} < 0$, then,

$$|\beta_{sta}| < \frac{|n_{\delta r}\delta_r|}{n_\beta - \alpha_0 l_\beta} \quad \text{for } \alpha_0 \neq 0 \qquad (61)$$

Standard Airframe Model Construction

Under the restriction of equations (46), (60) and (61), we assume the basic characteristic values of pitch and yaw dynamics of our standard airframe model as shown in Table1. Table 1 means the characteristic values at zero angle of attack, except for the notation ($\cdot$) in yaw dynamics, which means the characteristic values at 20 degrees of angle of attack. Namely, pitch characteristic values are assumed to change only by missile velocity. But in the case of yaw dynamics, those characteristic values are assumed to be the functions of both missile velocity and angle of attack. The effects of gravity on the yaw characteristic values are assumed to be negligible, therefore, these characteristic values shown in table 1 correspond to the case of zero bank angle (Φ=0) in equation (28). Here, by applying the characteristic values defined in table 1 to equations (42)-(45) for pitch dynamics and equations (55)-(59) for yaw dynamics, the aerodynamic stability derivatives used in equation (26)~(31) are obtained. The relations between control inputs vs. angle of attack, side slip angle and stationary acceleration multiple are summarized in Table2. Also. the sign of aerodynamic stability derivatives are in Table3.

Table1 Pitch and yaw dynamics basic characteristics

Flight Condition		30Kft 1.0M		30Kft 2.0M		30Kft 3.0M	
velocity (m/s)		304		608		912	
Pitch	ωn/2π (Hz)	0.5		1.25		2.0	
	ζ	0.2		0.2		0.2	
	α sta (deg)	10.0		8.0		6.0	
	n_z sta (g)	5.0		22.5		40.0	
Yaw	ωn/2π (Hz)	1.0	(1.2)	1.75	(2.1)	2.5	(3.0)
	ζ	0.08	(0.067)	0.08	(0.067)	0.08	(0.067)
	β sta (deg)	10.0	(3)	8.0	(2.4)	6.0	(1.8)
	n_y sta	4.0	(2.0)	12.0	(6.0)	19.0	(9.5)

Table 2 Response for control input

Pitch	$\delta e > 0$	$\alpha < 0$	nz > 0
	$\delta e < 0$	$\alpha > 0$	nz < 0
Yaw	$\delta r > 0$	$\beta > 0$	ny < 0
	$\delta r < 0$	$\beta < 0$	ny > 0
Roll	$\delta a > 0$	p > 0	—
	$\delta a < 0$	p < 0	—

Table 3 Signs of aerodynamic stability derivatives

Pitch	$m_\alpha < 0, z_\alpha < 0$	$m_{\delta e} < 0, z_{\delta e} < 0$
Yaw	$n_\beta > 0, y_\beta < 0$	$n_{\delta r} < 0, y_{\delta r} > 0$ $n_{\delta a} > 0, y_{\delta a} < 0$
Roll	$l_\beta < 0$	$l_{\delta a} > 0, l_{\delta r} < 0$

Decision of Pitch Dynamics Stability Derivatives

By using the data of the case of medium speed in Table 1 as an example, pitch dynamics stability derivatives m_α, Z_α, $m_{\delta e}$ and $Z_{\delta e}$ are obtained as follows.

In equation (42), by setting the natural frequency at 1.25Hz,

$$\omega_n = \sqrt{-m_\alpha} = 2 \times \pi \times 1.25$$

$$m_\alpha = -61.69 \qquad (62)$$

In equation (43), by setting the damping ratio at 0.2,

$$\varsigma = \frac{-z_\alpha}{2\sqrt{-m_\alpha}} = 0.2$$

$$z_\alpha = -3.14 \qquad (63)$$

In equation(44), by setting the stationary value of angle of attack at -8degrees, when the control input δ e equals to the full control deflection, 10 degrees,

$$\alpha_{sta} = \frac{m_{\delta e}}{-m_\alpha}\frac{10}{57.3} = \frac{-8}{57.3}$$

$$m_{\delta e} = -49.35 \qquad (64)$$

In equation (45), by setting the z-axis stationary acceleration multiple at 22.5, when the control input δ e equals to 10degrees,

$$n_{zsta} = \frac{V_m}{g}\left(\frac{z_\alpha m_{\delta e}}{-m_\alpha} + z_{\delta e}\right)\frac{10}{57.3} = 22.5$$

$$z_{\delta e} = -0.43 \qquad (65)$$

As same to above, we can obtain the stability derivatives for the flight conditions of low speed and high speed as shown in Table 4.

Table 4 Pitch stability derivatives

Flight Condition	30Kft 1.0M	30Kft 2.0M	30Kft 3.0M
m_α	-9.87	-61.69	-157.91
Z_α	-1.26	-3.14	-5.03
$m_{\delta e}$	-9.87	-49.35	-94.75
$Z_{\delta e}$	-0.34	-0.43	-0.56

Decision of Yaw Dynamics Stability Derivatives

By using the data of the case of medium speed in Table 1 as an example, yaw dynamics stability derivatives n_β, y_β, $n_{\delta r}$ and $y_{\delta r}$ are obtained as follows, when angle of attack equals to zero and the effects of gravity is ignored. In equation (55), by setting the natural frequency at 1.75 Hz ,

$$\omega_n = \sqrt{n_\beta} = 2 \times \pi \times 1.75$$

$$n_\beta = 120.9 \qquad (66)$$

In equation (56), by setting the damping ratio at 0.08,

$$\varsigma = \frac{-y_\beta}{2\sqrt{n_\beta}} = 0.08$$

$$y_\beta = -1.76 \qquad (67)$$

In equation (57), by setting the stationary value of side slip angle at 8 degrees, when the control input δ r equals to the full control deflection, 5 degrees.

$$\beta_{sta} = -\frac{n_{\delta r}}{n_\beta}\frac{5}{57.3} = \frac{8}{57.3}$$

$$n_{\delta r} = -193.44 \qquad (68)$$

In equation (59), by setting the y-axis stationary acceleration multiple at 12, when the control input δ r equals to –5 degrees,

$$n_{vsta} = \frac{V_m}{g}\left(\frac{y_\beta n_{\delta r}}{-n_\beta} + y_{\delta r}\right)\cdot\frac{-5}{57.3} = 12$$

$$y_{\delta r} = 0.60 \qquad (69)$$

As same to above, we can obtain the stability derivatives for the flight conditions of low speed and high speed as shown in Table 5.

Table 5 Yaw stability derivatives(α_0=0)

Flight Condition	30Kft, 1.0M	30Kft, 2.0M	30Kft, 3.0M
n_β	39.48	120.90	246.74
y_β	-1.01	-1.76	-2.51
$n_{\delta r}$	-78.96	-193.44	-296.09
$y_{\delta r}$	0.54	0.60	0.67

Decision of Yaw-Roll Coupling Stability Derivatives l_β

The natural frequency of yaw dynamics when angle of attack exists is given by equation (55).

$$\omega_n = \sqrt{n_\beta - \alpha_0 l_\beta} \qquad (55)$$

Here, assuming that, the variation of the natural frequency in yaw dynamics is quite caused by l_β, and l_β is constant for all angle of attack, then by using the assumption of natural frequency for 20 degrees of angle of attack shown in Table 1, the derivative l_β can be obtained as shown in Table 6.

Table 6 Yaw-Roll Coupling Stability Derivative l_β

Flight Condition	l_β
30Kft, 1.0M	-49.76
30Kft, 2.0M	-152.42
30Kft, 3.0M	-311.04

Yaw-Roll Dynamics Stability Derivatives with Angle of Attack

In the yaw dynamics stability derivatives shown in table 5, it can be assumed that n_β and y_β are constant for the change of angle of attack, but generally, $y_{\delta r}$ and $n_{\delta r}$ decrease according to the growing angle of attack. The values of $y_{\delta r}$, $n_{\delta r}$ and $l_{\delta r}$ when angle of attack equals to 20 degrees can't be determined uniquely from the assumption of Table 1, because, Table 1 gives only two equations for three unknown parameters of $y_{\delta r}$, $n_{\delta r}$ and $l_{\delta r}$, namely equation (58) and (59). Therefore, here, we assume that $y_{\delta r}$ decreases linearly against the increase of angle of attack, and at 20 degrees of angle of attack, it becomes half value of zero angle of attack. Then, from equation (59), $n_{\delta r}$ for 20 degrees of angle of attack is obtained under the assumption of acceleration multiple shown in Table 1, and it can be linearly distributed to other angle of attack. And $l_{\delta r}$ for 20 degrees angle of attack is also obtained from equation (58) under the assumption of side-slip angle. On the other hand, the effect of roll control input δa to yaw dynamics increases as growing angle of attack, therefore, we assume the value of $y_{\delta a}$ and $n_{\delta a}$ at 20 degrees of angle of attack as half of $y_{\delta r}$ and $n_{\delta r}$, and by distributing the value linearly to angle of attack, Table 7, 8 and 9 are obtained. Of course, $y_{\delta a}$ and $n_{\delta a}$ can be assumed to be zero at zero angle of attack for up and down symmetry airframe. $l_{\delta r}$ obtained here for 20 degrees angle of attack is used in the section of roll dynamics stability derivatives.

Decision of Roll Dynamics Stability Derivatives

The rest of undefined stability derivatives are $l_{\delta a}$ and $l_{\delta r}$. The value of $l_{\delta r}$ for 20 degrees of angle of attack is already obtained from the assumption of side slip angle, and $l_{\delta r}$ can be assumed to be zero at zero angle of attack for this up and down symmetry airframe. Then by distributing linearly to angle of attack, $l_{\delta r}$ can be decided as shown in Table 10,11 and 12. On the other hand, the stationary values of roll rate p for step δa and δr control inputs under the existence of gravity effects are given as equation (70) and (71) by use of equation (51).

$$p_{sta}|_{\delta a} = \frac{v_m}{g}\frac{n_{\delta a} l_\beta - l_{\delta a} n_\beta}{l_\beta}\delta_a \qquad (70)$$

$$p_{sta}|_{\delta r} = \frac{v_m}{g}\frac{n_{\delta r}l_\beta - l_{\delta r}n_\beta}{l_\beta}\delta_r \qquad (71)$$

In our airframe model, the maximum deflection of δ a and δ r are same values of 5 degrees. Generally, it is natural that the effect of control input δ a on role rate p is greater than δ r 's effect for the same deflection of control input, the relation of equation (72) must be satisfied for all angle of attack.

$$l_{\delta a} + l_{\delta r} > \frac{l_\beta}{n_\beta}(n_{\delta a} + n_{\delta r}) \qquad (72)$$

Now, all the stability derivatives except for $l_{\delta a}$ in equation (72) are already obtained for all angle of attack, the minimum value of $l_{\delta a}$ can be decided from equation (72). But here in this airframe model, we decide the value of $l_{\delta a}$ under the condition of equation (73) for all angle of attack.

$$P_{sta}|_{\delta a=\delta a\,max} = -1.5 * P_{sta}|_{\delta r=\delta r\,max} \qquad (73)$$

In equation (73), the minus sign means the opposite effects of δ a and δ r on roll rate p, and the multiple 1.5 is selected to ensure the predominance of δ a on roll dynamics for all angle of attack. Then, $l_{\delta a}$ is decided from equation (74) for each angle of attack. The results of $l_{\delta a}$ and $l_{\delta r}$ are shown in Table 7,8 and 9.

$$l_{\delta a} = \frac{l_\beta}{n_\beta}(n_{\delta a} + 1.5n_{\delta r}) - 1.5l_{\delta r} \qquad (74)$$

Under the values of $l_{\delta a}$ and $l_{\delta r}$ shown in Table 7,8 and 9, roll control input δ a is always ensured the predominance compared to δ r in roll dynamics.

Standard Airframe Models

All the aerodynamic stability derivatives that are needed for the standard airframe model equations (26) ~(31) are obtained now. Those stability derivatives are summarized in Table 7,8 and 9.

Table 7 Standard Airframe Model Stability Derivatives 30Kft, 1.0M

α (deg)	0	5	10	15	20
Z_α	-1.26	-1.26	-1.26	-1.26	-1.26
$Z_{\delta e}$	-0.34	-0.34	-0.34	-0.34	-0.34
m_α	-9.87	-9.87	-9.87	-9.87	-9.87
$m_{\delta e}$	-9.87	-9.87	-9.87	-9.87	-9.87
y_β	-1.01	-1.01	-1.01	-1.01	-1.01
$y_{\delta r}$	0.54	0.47	0.41	0.34	0.27
$y_{\delta a}$	0	-0.03	-0.07	-0.10	-0.14
n_β	39.48	39.48	39.48	39.48	39.48
$n_{\delta r}$	-78.96	-69.08	-59.20	-49.32	-39.44
$n_{\delta a}$	0	4.93	9.86	14.79	19.72
l_β	-49.76	-49.76	-49.76	-49.76	-49.76
$l_{\delta r}$	0	-3.82	-7.64	-11.45	-15.27
$l_{\delta a}$	149.23	130.08	110.92	91.75	72.60

Table 8 Standard Airframe Model Stability Derivatives 30Kft,2.0M

α (deg)	0	5	10	15	20
Z_α	-3.14	-3.14	-3.14	-3.14	-3.14
$Z_{\delta e}$	-0.43	-0.43	-0.43	-0.43	-0.43
m_α	-61.69	-61.69	-61.69	-61.69	-61.69
$m_{\delta e}$	-49.35	-49.35	-49.35	-49.35	-49.35
y_β	-1.76	-1.76	-1.76	-1.76	-1.76
$y_{\delta r}$	0.60	0.53	0.45	0.38	0.30
$y_{\delta a}$	0	-0.04	-0.08	-0.11	-0.15
n_β	120.90	120.90	120.90	120.90	120.90
$n_{\delta r}$	-193.44	-169.27	-145.09	-120.92	-96.74
$n_{\delta a}$	0	12.09	24.19	36.28	48.37
l_β	-152.42	-152.42	-152.42	-152.42	-152.42
$l_{\delta r}$	0	-9.43	-18.87	-28.30	-37.73
$l_{\delta a}$	365.89	319.07	272.24	225.42	178.58

Table 9 Standard Airframe Model Stability Derivatives 30Kft,3.0M

α (deg)	0	5	10	15	20
Z_α	-5.03	-5.03	-5.03	-5.03	-5.03
$Z_{\delta e}$	-0.56	-0.56	-0.56	-0.56	-0.56
m_α	-157.91	-157.91	-157.91	-157.91	-157.91
$m_{\delta e}$	-94.75	-94.75	-94.75	-94.75	-94.75
y_β	-2.51	-2.51	-2.51	-2.51	-2.51
$y_{\delta r}$	0.67	0.59	0.51	0.42	0.34
$y_{\delta a}$	0	-0.04	-0.09	-0.13	-0.17
n_β	246.74	246.74	246.74	246.74	246.74
$n_{\delta r}$	-296.09	-259.17	-222.26	-185.34	-148.42
$n_{\delta a}$	0	18.55	37.11	55.66	74.21
l_β	-311.04	-311.04	-311.04	-311.04	-311.04
$l_{\delta r}$	0	-14.69	-29.38	-44.07	-58.76
$l_{\delta a}$	560.05	488.86	417.68	346.49	275.30

Actuator Model

The actuator for fin control can be modeled simply by the block diagram shown in Fig.6. This model simply consists of a saturation element and a second order system. The characteristic values are, K=89.76. T=0.0057. This means that the actuator has the characteristics of natural frequency 20Hz. and damping ratio 0.7. Yaw and roll

control actuator is same one and the total maximum deflection of the actuator is 10 degrees.

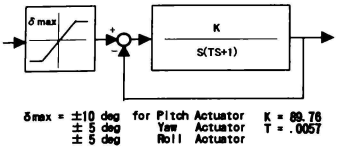


Fig.6 Actuator Model

Conclusion

A standard air-to-air missile airframe model which is inevitably needed for the studies on robust autopilot system design and/or evaluation has been proposed in this paper. Although this proposed model doesn't have certain dimension in actual, it has the characteristics of a realistic short/ medium range air-to-air missile. By using this universal airframe model as a control object, it becomes possible to evaluate missile robust autopilot system design and/or evaluation methods fairly.

REFERENCES

[1]Lin.C.F.,Lee,S.P.,"Robust Missile Autopilot Design Using a Generalized Singular Optimal Control Technique", Journal of Guidance, Control, and Dynamics, Vol.8, No.4, 1985, pp498-507.

[2]Williams,D.E. et al., "Modern Control Theory for Design of Autopilots fir Bank-to-Turn Missiles," Journal of Guidance, Control, and Dynamics, Vol.10, No.4, 1987,pp.378-386.

[3]Arrow,A., Williams,D.E., "Comparison of Classical and Modern Missile Autopilots Design and Analysis Technique", Journal of Guidance, Control, and Dynamics, Vol.12, No.2,1989,pp220-227.

[4]Wise.K.A.,"Bank-to-Turn Missile Autopilot Design Using Loop Transfer Recovery", Vol.13, No.1,1990,pp145-152.

[5]Wise,K.A., "Singular Value Robustness Tests for Missile Autopilot Uncertainty", Journal of Guidance, Control, and Dynamics, Vol.14, No.3,1991,pp597-606.

[6]Keel,L.H., et al., "Robust Eigenvalue Assignment with Maximum Tolerance to System Uncertainties", Journal of Guidance,Control,and Dynamics,Vol.14,No.3,1991,pp615-620.

[7]Green,A., et al., "Game Optimal Guidance Law Synthesis for Short Range Missiles", Journal of Guidance, Control, and Dynamics, Vol.15, No.1,1992,pp191-197.

[8]Wise, K.A., "Comparison of Six Robustness Tests Evaluating Missile Autopilot Robustness to Uncertain Aerodynamics", Journal of Guidance, Control, and Dynamics, Vol.15, No.4, 1992, pp861-870.

[9]Li,X.P., et al., "Robust Control System Design Using H^{∞} Optimization Theory", Journal of Guidance, Control, and Dynamics, Vol.15, No.4,1992,pp944-952.

[10]Wise,B., Bern,D.S., "Bench Mark Problem for Robust Control Design", Journal of Guidance, Control and Dynamics, Vol.15, No.5, 1992, pp 1057-1059.

[11]Wise,K.A., "Missile Autopilot Robustness Using the Real Multi-Loop Stability Margin", Journal of Guidance, Control, and Dynamics, Vol.16, No.2,1993,pp354-362.

[12]Brumbaugh,R.W., "Aircraft Model for the AIAA Controls Design Challenge". Journal of Guidance, Control and Dynamics, Vol.17, No.4, 1994, pp747-752.

[13]Lin,C.F., et al., "High Performance Robust Bank-to-Turn Missile Autopilot Design", Journal of Guidance, Control, and Dynamics,Vol.18,No.1,1995,pp46-53.

[14]Apkarian,P., "Self-Scheduled H ∞ Control of Missile via Linear Matrix Inequalities", Journal of Guidance, Control, and Dynamics, Vol.18, No.3, 1995,pp532-538.

[15]Cho,H., et al., "Closed-Form Optical Guidance Laws for Missiles of Time-Varying Velocity", Journal of Guidance, Control, and Dynamics, Vol.19, No.5, 1996,pp1017-1022.

[16]Carter,L.H., et al., "Gain-Scheduled Bank-to-Turn Autopilot Design Using Linear Parameter Varying Transformations", Journal of Guidance, Control, and Dynamics, Vol.19, No.5, 1996, pp1056-1063.

[17]Fu,L.C., et al., "Adaptive Robust Bank-to-Turn Missile Autopilot Design Using Neural Networks", Journal of Guidance, Control, and Dynamics, Vol.20, No.2, 1997,pp346-354.

HARDWARE IN THE LOOP SIMULATION OF AN ATTITUDE CONTROL SYSTEM

Waldemar C. Leite Filho[†], Lais M. R. Mallaco[†] and Domingos S. Carrijo *
Instituto de Aeronáutica e Espaço,
12228-904 - S. José dos Campos (SP) Brazil

ABSTRACT

In this work, part of the hardware in the loop simulation of the Brazilian Satellite Launcher VLS attitude control system is described. This particular attitude control system aims to point the last stage of the launch vehicle, prior to its ignition, in order to inject a satellite in a specified orbit. The main result is the final positioning accuracy of the pointing system, which matched the specification. The detailed simulation scheme is presented as well as the simulation results.

INTRODUCTION

The complexity of aerospace vehicle control systems and the high cost of even simple flight tests lead the application of simulation in aerospace vehicle control design to be extensive and absolutely necessary.

Hardware in the Loop Simulation (HWIL)[1], also named hybrid simulation, is the standard test bed for assessing the flight worthiness of the guidance, navigation and control software and hardware, in a real time environment, what helps debugging system problems and permits the comparison between the performance forecasted in the design of the control system and the performance obtained when the actual elements are working together. HWIL simulation provides detection and correction of design deficiencies, prior to the flight. A HWIL simulation should include all possible real parts of the system in order to verify the correct implementation of the control system design, check its performance and help the validation of the experimental models.

In this work, part of the hardware in the loop simulation of the Brazilian Satellite Launcher VLS attitude control system is described. The attitude control system considered aims[2] to point the last stage of the launch vehicle to a prescribed inertial direction, prior to its ignition, in order to inject a satellite in a specified orbit. Since all the angular errors of the vehicle will be transferred to the satellite orbit, the performance of this control system must be checked very carefully. Thus, it is necessary for the control loop to be as close as possible of the real situation, without any digital simulation.

The process to be tested is the control system of a rotating vehicle, which uses on-off actuators (cold gas system). This implies the existence of a limit-cycle and of a coupling between maneuver plans. The dynamic of this process is complex enough to impair the use of linear techniques of analysis. So the best option is a simulation test where the real components _ controller, actuators and sensors _ are present and are qualified based on their global performance.

To include the real vehicle dynamics in the simulation, it is necessary to have a rotational motion. This is possible when the vehicle is assembled over an air-bearing device. The assembly of the test bench was done with a mock-up of the VLS last stage including the satellite. In this mock-up the complete real control system was implemented (see Picture 1).

EXPERIMENT DESCRIPTION

The design of this attitude control system is described by Leite Filho & Mallaco[3]. The controller action is divided in four phases to avoid the coupling between the planes pitch, yaw and roll. Since the real motion will take place out of the atmosphere, the vehicle dynamics can be considered as a "torque-free motion"[4] except for the control torque.

The four phases are[3]: **1st** - Stopping of the residual rotational motion, what means reducing the angular rates p, q and r to their minimums to reduce the motion coupling between the three planes; **2nd** - Turn the vehicle around its roll axis in order to drive the roll angle Φ to a pre-calculated value such that, at the end of this phase, the plane where the pitch valves are located coincides with the plane that permits the vehicle to maneuver straight to the correct attitude (θ_p . ψ_p); **3rd** - Pitch-over maneuver in pitch plane (set at the end of 2nd phase) to reduce the errors θ_p - θ and ψ_p - ψ, expressed in the body frame, to their minimums; **4th** - Maintain the vehicle pointed (in limit-cycle) until the command of spin-up.

[†] Senior Researcher of Space System Dept., AIAA Associate Member

[‡] * Senior Researchers of Space System Dept.

Copyright©1999 by American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

The mock-up assembled over the air-bearing bench includes: on-board computer, angular sensor, rate gyros as well as the actuators (the complete on-off cold gas system).

All physical parameters like inertia, center of mass (moment arm) and thrust were estimated to be the same of the real flight. With these conditions it was possible to reproduce the expected motion of the real flight and so, tune the controller gains and dead zones, evaluate the effects of sensors noise; evaluate the effects of actuators delay; evaluate the effects of the sliding mode and the characteristics of the limit-cycle, validate the software implementation[5] and the performance of the actuation system. The sampling frequency of the controller (on-board computer) is 64 Hz.

Tests were fulfilled considering different angular positions and velocities as initial conditions. The control system is started and must drive the vehicle to the specified angular position (through the four phases), keeping it there during one hundred seconds with an accuracy better than 0.1°.

RESULTS

From among several tests fulfilled, there are the ones shown in the figures 1 and 2. Fig. 1 shows the pitch plane motion: (a) phase plane q x θ; (b) attitude time history θ (t) and (c) angular rate time history q (t). Fig. 2 shows the roll plane motion: (a) phase plane p x ϕ; (b) attitude time history ϕ (t) and (c) angular rate time history p(t). The time histories of both figures show the comparison with the digital simulation for each case.

The 2nd phase (BASC2)[3] is present only in fig. 2, since it concerns roll maneuver. The 3rd phase (BASC3)[3] is present only in fig. 1, since it concerns pitch maneuver.

Both cases (fig.1 and 2) show that the optimal solution was not performed exactly. Optimal solution is proposed in phase 2 and 3 and would lead to zero using only one commutation (phase plane) or no overshoot (angular time history). This fact can be explained by the decreasing of the inertia (loss of gas mass) during the test vis-à-vis the fixed coefficients used in the control algorithm. Moreover, some difference between the actual force of the thrust present in the test and that one measured in the laboratory (thrust valves) was detected. It was found that a turbulence in the gas flow decrease its efficiency.

During the 4th phase (BASC4)[3] the controlled system exhibit a limit cycle. Its frequency is given by the combination of the time-delays of the on-off valves, the flowing gas dynamics and the controller characteristics (filters and dead-zones). Fig. 3a shows the time history of the angular rate q(t) where the limit cycle can be observed. Angular rate was chosen because of its noise level. Besides that, fig. 3b shows its frequency spectrum (from FFT) where the value of the limit cycle can be estimated (1.8 Hz). It is expected that this frequency is time varying because of the decreasing of the gas mass and, consequently, the decreasing of the inertia.

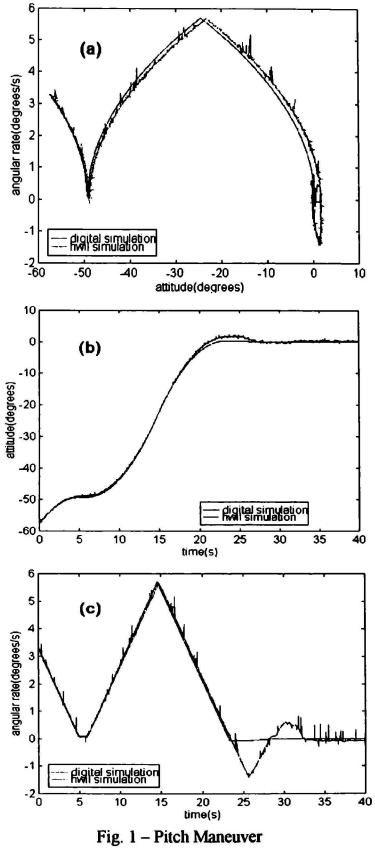


Fig. 1 – Pitch Maneuver

The final errors depend not only on the control loop characteristics (as dead zone, sampling frequency and limit cycle amplitude) and characteristics of the actuators (as level of thrust and time delay in the

opening and shutting), but it also depends on the accuracy of the sensors, its noise level and the quantization (ADC with 12 bits).

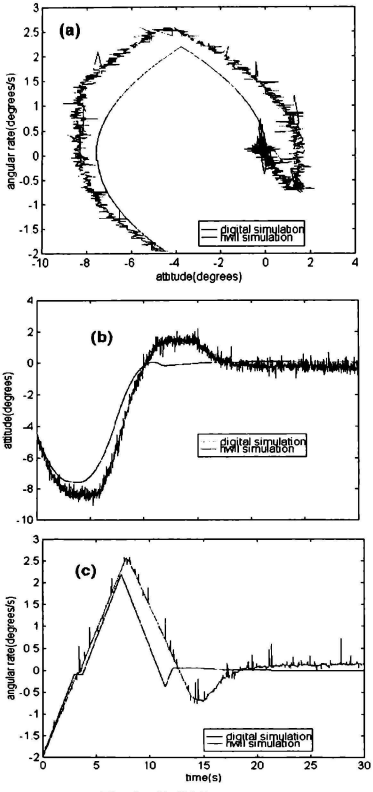


Fig. 2 – Roll Maneuver

Due to these facts, the final values of the angular rate and the pointing angle will be taken as the average values of the acquired measurements in the last 10 seconds of each test. The following values were found: Fig.1 - angular error = -.11° , angular rate = -.05°/s and Fig. 2 - angular error = .23°, angular rate = .13°/s.

The final accuracy of the roll angle and of the roll angular rate is not fundamental. The acceptable minimum value of 0.01°/s for the angular rate p was defined in order to reduce the coupling during the pitch maneuver. The reduction of the angle ϕ to a value smaller than 0.1° has the objective of optimizing the use of the stored energy, by matching the actuation plane (plane of valves) with the direction of the wanted motion. These acceptable errors were stated as dead-zones of ± 0.1° in the controller algorithm.

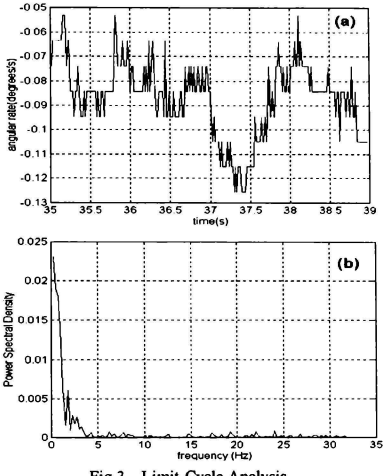


Fig.3 – Limit-Cycle Analysis

COMMENTS AND CONCLUSION

The whole simulation test was considered successful. Failures were detected and fixed. Algorithms were checked and some coefficients were tuned. The main result is the final accuracy of the pointing system, which matched the specification.

The comparison between the HWIL simulation results and the all-digital simulation results provides more fidelity to the mathematical models used. With this test it was also possible to check several safe modes like dispersion cases and the total gas consumption.

REFERENCES

[1] Leite Filho, W.C. & Carrijo, D.S., "Hardware in The Loop Simulation of Brazilian Launcher VLS", *Proceedings of 3rd ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*, Noordwijk, The Netherlands, Nov. 1996, pp. 355-358.

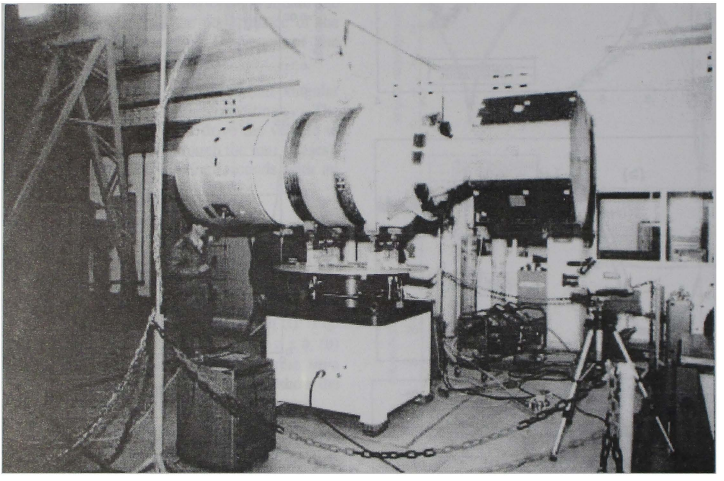
[2] Leite Filho, W.C. and Pinto, P.S., "Guidance Strategy for Solid Propelled Launchers", AIAA *Journal of*

Guidance, Control and Dynamics, Vol. 21, No. 6, 1998, pp. 1006-1009.

[3] Leite Filho, W.C. & Mallaco, L.M.R., "Strategy for Pitch-Over Maneuver Control", *AIAA Paper No. 99-4049*, Aug. 1999.

[4] Wie, B., "Rigid-Body Dynamics", *Space Vehicle Dynamics & Control*, AIAA, Reston, VA, 1998, pp. 331-380.

[5] Carrijo, D.S., Oliva, A.P. & Leite Filho, W.C., "Hybrid Simulation Software Scheme for Validation of a Launcher Control System", *Preprints of 14th IFAC Symposium on Automatic Control in Aerospace*, Seoul, Korea, Aug. 1998, pp. 427-432.



Picture 1 – Control System in Hardware in the Loop Simulation

GROUND EFFECT MODELING USING A HYBRID APPROACH OF INVERSE SIMULATION AND SYSTEM IDENTIFICATION

D. Fischenberg

Institut für Flugmechanik
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Lilienthalplatz 7, D-38108 Braunschweig
Germany

Abstract

A 2-point aerodynamic model for ground effect influence in the longitudinal motion is presented. The model structure is derived using flight test data and inverse simulation techniques. The model basically provides five parameters, two for the analytical approximation of the ground effect influence function stated by *Prandtl* and three parameters to quantify the main aerodynamic changes in ground effect: increase of the lift slope, decrease of the induced drag, and reduction of the wing induced downwash angle. The model includes the aerodynamic lag of this downwash angle to be delayed at the horizontal tail. The model parameters are identified from VFW-614 flight test data. The results are validated by checking the physical plausibility and compared to theory for different flap settings.

Nomenclature

a	coefficient of the ground effect influence function
α	angle of attack
b	wing span
C_L	total A/C lift coefficient
$C_{L\alpha}$	lift slope
C_D	total A/C drag coefficient
C_{Di}	induced drag
D	drag
Δh_{ra}	radar altimeter measurement offset
Δt	time delay: r_t^*/V
δ_e	elevator deflection
e	Oswald-factor
ϵ_t	downwash angle at horizontal tail
h	altitude above ground
h_{ra}	altitude measured by radar altimeter
i_t	horizontal tail trim angle
L	lift
Λ	aspect ratio
Θ	pitch angle
S	wing reference area
S_t	horizontal tail reference area
σ	ground effect influence function
r_t	distance from CG to NP horizontal tail
r_t^*	distance from NP wing to NP horizontal tail
t	time
u	control input
V	true airspeed
y	model output
x, z	distances from CG to radar altimeter antenna

A/C	aircraft
CG	center of gravity
FAA	Federal Aviation Administration
ge	ground effect
MAC	mean aerodynamic chord
NP	neutral point
ref	reference
t	horizontal tail
wb	wing-body
0	zero (coefficient)

Introduction

Takeoff and landing are important areas for pilot training on flight simulators. The quality of a simulator can be characterised by the capability of its moving, visual, and sound systems and the realism of the mathematical model. The quality, mainly that of the mathematical model, is specified within the internationally accepted ICAO approval standard, which defines different quality levels. The national standards are mostly identical with the ICAO standard, e. g. the Advisory Circular for Airplane Simulator Qualification of the FAA [1].

Operating on and near the ground, the aircraft aerodynamics is influenced by the proximity of the ground, which is known as 'ground effect'. This ground effect has considerable influence on e. g. takeoff performance

Copyright ©1999 by DLR Institut für Flugmechanik. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

and flare characteristics. Ground effect influence depends on aircraft design, e. g. it is stronger on low wing configurations than on top wing ones, also the horizontal tail position is an important factor.

For approval according to a certain quality level, the simulator model has to be validated with flight test data. The model is stimulated using the flight test measured pilot inputs, its response is compared to the flight measured aircraft response. Besides takeoff and landing maneuvers, additional flight tests are required to qualify the ground effect aircraft model: level flybys in about 150%, 70%, 30%, and 10% wingspan altitude. The allowable tolerances (differences between model output and aircraft measured data) are 1° pitch, angle of attack, and elevator, 3 *kt* airspeed, 5% power for level flight and 5 *ft* altitude.

Ground effect influence can be determined theoretically or can be measured in wind tunnel. A lot of work is available on these topics. Only a few basic investigations may be pointed out: both *Prandtl* and *Wieselsberger* compared theoretical computations to (simple) wind tunnel measurements in 1923 [2], [3]. *Lippisch* collected a lot of ground effect wind tunnel measurements for discussion in 1964 [4], and *Carter* made some interesting measurements in 1961 [5].

The quality of ground effect models is often not sufficient for high fidelity simulators. In this paper a method is described, in which a ground effect model is developed evaluating flight test data with maximum likelihood parameter identification techniques [6]. The model determination will be performed in two steps: (a) separation of ground effect influence with inverse simulation techniques [7], and (b) model parameter estimation. Flight test data of the twin-turbofan aircraft VFW-614 will be used. A flight test validated aerodynamic 2-point model [8], which describes the forces of the wing and the tail separately [9], is used as a basis for adding the ground effect model in an incremental way.

Flight Test Program

Special flight tests have to be conducted for ground effect evaluation. An important requirement for the test condition is a calm day with low horizontal wind and very low turbulence. Otherwise it is hardly possible to separate ground effect influence [10].

The flight test aircraft used for this evaluation is the DLR research aircraft VFW-614, which is a low wing configuration with two turbofan engines (Fig. 1). It is equipped with a complete onboard measurement system including a noseboom and angle of attack and sideslip vanes. The flight test data are stored onboard.

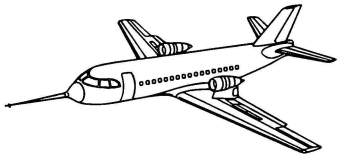


Figure 1: Test aircraft VFW-614

The test program consisted of tower fly-bys and touch and gos with an extensive flare. For each flap setting, 4 stationary fly-bys, 4 fly-bys with small superimposed elevator, aileron and rudder doublets and two touch and go maneuvers were flown. The fly-bys lasted about 30 s, the airspeed was about 1.3 V_s in the gear down position. The level fly-bys were spread in altitude between 100 ft (outside ground effect) and 10 ft in 150%, 70%, 30%, and 10% wingspan altitude.

Before each fly-by, there was an actual wind measurement from the tower. The wind was measured between 2...5 *kt* horizontally for most of the tests, only during a very few tests the wind grew up to 8 *kt*. During evaluation of the flight test data, these conditions proved to be tolerable for separation of longitudinal ground effect, but lateral ground effect influence was very hard to see. This may originate from a certain level of turbulence, that is obvious in the flight test data.

Measurements

The test aircraft is equipped with a usual flight test intrumentation. The same instrumentation was formerly used for identification of the basic model and determination of the aerodynamic landing gear effects. It may be pointed out, that, for incremental model identification, only flight test data should be evaluated, that are recorded in a limited period of time with one aircraft and the same test instrumentation.

Special attention has to be payed to the radar altimeter, that is important for exact determination of altitude above ground. The measurement is of limited resolution, in this case 1 ft, which proved to be sufficient for the evaluation. Additionally, a measurement bias is evident, that can be determined while the aircraft is standing statically on the runway. This bias should be checked before and after each flight, as it may have a drift and no check is possible during fly-by maneuvers.

The ground effect strongly depends on altitude and should be referenced to a fixed geometric aircraft point. As the aerodynamic changes are mostly induced by the wing, a good point is the 25% MAC point on the airfoil camber line. Using the measured radar altitude as model input, the altitude of this reference point $h_{ge,ref}$ can be determined as

$$h_{ge,ref} = h_{ra} - x \cdot \sin\Theta + z\cos\Theta - \Delta h_{ra}, \quad (1)$$

whereas h_{ra} is the measured radar altitude, x and z are the bodyfixed distances between the radar altimeter antenna and the reference point, Θ is the aircraft pitch angle, and Δh_{ra} is the measurement bias.

It should be mentioned that the angle of attack measurement proved to be influenced by the ground effect, too, and therefore should be looked at carefully. Additionally, it is helpful to record the undercarriage status for detection of the touch down point.

Basic Ground Effect Physics

Ground effect influence is strongly dependent on the considered airfoil, wing geometry or aircraft configuration. But a few main effects are generally apparent and will be discussed now.

Basically spoken, the main effect on a finite wing in the proximity of the ground is a reduction of the wing tip vortices which is equivalent to a reduction of the induced angle of attack. That means, for a constant geometrical positive angle of attack the lift coefficient will increase if the altitude above ground decreases. Wind tunnel measurements, e. g. by *Fink* [11] or *Carter* [5], confirm a considerable increase in the lift curve *slope* in the proximity of the ground.

The reduction of the induced angle of attack results consequently in the reduction of the induced drag. This was the main motivation for the development of ground effect vehicles or 'aerofoil-ships', see e. g. [4]. According to *Wieselsberger's* theory [3], the induced drag (for a wing with elliptical lift distribution) can be expressed as a function of σ, the ground effect influence function:

$$C_{Di} = (1-\sigma)\frac{C_L^2}{e\pi\Lambda}. \quad (2)$$

σ is a nonlinear function of h/b (altitude/wing span) and varies between $0...\sigma...1$. For $h/b = 0$, the induced drag will completely disappear. *Prandtl* [2] gave some values for the ground effect influence funktion σ, see *Fig. 2.* Three important values of σ are collected in the following table.

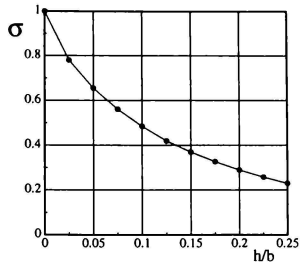


Figure 2: Ground effect influence function acc. to Prandtl [2]

h/b	∞	0.1	0
σ	0	0.5	1

Assuming a quadratic polar, the L/D-increase in ground effect can simply be expressed as [4]:

$$\frac{\left(\frac{L}{D}\right)_{ground}}{\left(\frac{L}{D}\right)_{h/b=\infty}} = \frac{1}{\sqrt{1-\sigma}}. \quad (3)$$

Taking $h/b = 0.1$, the L/D-increase in ground effect would be $\frac{1}{\sqrt{1-\sigma}} = \sqrt{2}$. These results were validated through several wind tunnel measurements, not only for airfoils, but even for complete aircraft configurations [3], [5].

The pitching moment is also influenced by the ground effect. First, the neutral point of the wing can be changed - in which direction and how much is strongly dependent on airfoil and angle of attack, see [11], [12]. The most important change in ground effect is the reduction of the wing induced downwash angle at the horizontal tail. As a consequence, horizontal tail angle of attack is increased, which results in additional tail lift and a pitch down moment for the complete aircraft.

Inverse Simulation - A Tool for Model Development

A first, but very helpful step for model structure determination is the application of inverse simulation techniques to flight test data.

The principle of the inverse simulation technique used here is also known as *simplified inverse simulation* [7] [13], see *Fig. 3.* The flight test measured pilot inputs u_{pilot} (elevator, aileron, rudder, thrust or equivalent)

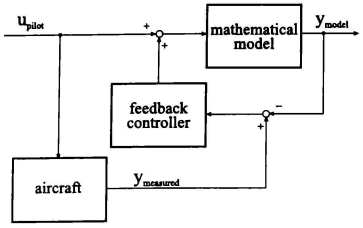


Figure 3: Principle of Simplified Inverse Simulation

are fed into the simulation model. The model output y_{model} is compared to the measured aircraft response $y_{measured}$, the difference (= model deficiencies) are the input into a feedback controller. The controller outputs are added to the measured pilot inputs. The result is that model deficiencies can no longer be seen in the output variables, but in the control variables.

The advantage of this method is that model deficiencies can be analysed more clearly, because the model is not leaving its trim point. Using an open loop simulation (without controller feedback), the differences between the model outputs and the measurements would increase rapidly. e. g. a wrong pitching moment would result very soon in a considerable variation of airspeed. In that case hardly any interpretation in terms of model structure development is possible.

As an example, the application of this method to the modeling of the change of pitching moment in ground effect will be discussed. Prerequisite for this case is a flight test validated simulation model including aerodynamic landing gear effects, but without ground effect model (the task is to develop this model). For this example the simulation altitude is set to the measured values. *Fig. 4* shows a VFW-614 landing approach and flare. The simulation model includes *no* ground effect, but the measured elevator is fed into the simulation. Only for the first 3 seconds the model is correct, thereafter the pilot inputs are no longer appropriate to fit the measured pitch angle. From this it is only possible to say that there is a certain amount of pitch down in ground effect, but no answer can be given how much and how it depends on altitude above ground.

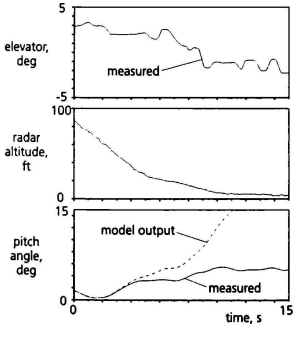


Figure 4: VFW-614 landing approach and flare *without* ground effect model

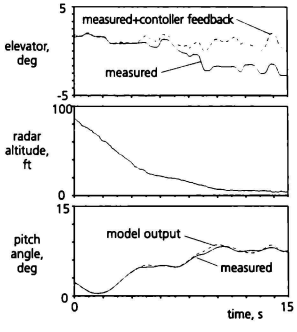


Figure 5: VFW-614 landing approach and flare *without* ground effect model, but with controller feedback

Fig. 5 shows the same maneuver, again no ground effect model, but with a controller feedback $\Delta\Theta \rightarrow$ *elevator* with a proportional and an integral part. Now the pitch angle is matched satisfactorily. The difference between the measured elevator and the model input is just the amount of controller activity. Now the progressive increase of the ground effect and its equivalent amount of elevator (about 4 deg at touch down) can be seen clearly. The task for model development is now to reduce this controller activity to zero.

Ground Effect Influence Function

From inverse simulation, the feedback activity as a function of altitude is a good basis for the shape of the ground effect influence function σ. From these evaluations, an analytical approximation of this functionality

can be achieved using the *tanh*-function:

$$\sigma = 1 - \tanh\left(a \cdot \frac{h}{b}\right) \tag{4}$$

Fig. 6 compares a parameter variation of a in eq. (4). to values stated by *Prandtl* [2]. The tanh function is

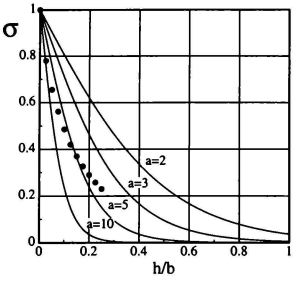


Figure 6: Parameter variation of *tanh*-approximation of ground effect influence function, compared to values given by Prandtl [2]

not an ideal, but rough approximation of the theory. The advantage of the function is that there is only one parameter (a) to adapt and no switching depending on altitude is necessary. As eq. (4) is steady for all altitudes, the *tanh* is very applicable to identification problems.

Aerodynamic 2-Point Ground Effect Model

- Longitudinal Motion -

For ground effect influence separation from flight test data, it is important to use a validated aerodynamic model as a basis. This basis model has to include the aerodynamic landing gear effects. From this model status, the ground effect influence can be added incrementally.

The basic VFW-614 model is a 2-point aerodynamic model [8]. The model describes fuselage/wing lift and horizontal tail lift separately. The advantage of this formulation is that several nonlinear effects, e. g. the downwash lag effect, can be modeled appropriately.

The now stated model is the result of several iterations of model structure determination. It is a compromise of describing the aerodynamic effects as precisely as possible, but using only a minimum of additional parameters.

Stripping the model to the essential effects, the lift coefficient for the wing-body section is modeled as:

$$C_{L,wb} = C_{L0} + (C_{L\alpha} + C_{L\alpha,ge} \cdot \sigma_1) \cdot \alpha, \tag{5}$$

with the index ge for ground effect. Horizontal tail lift is:

$$C_{L,t} = C_{L\alpha,t} \cdot \alpha_t + \frac{\partial C_{L,t}}{\partial \delta_e}\delta_e \tag{6}$$

The effective tail angle of attack α_t is the sum of wing angle of attack α, horizontal tail trim angle i_t, the wing downwash angle at the tail ϵ_t, and the dynamic angle of attack $\alpha_{dyn} = \tan^{-1} q r_t / V$ with r_t being the distance from the aircraft's center of gravity to the horizontal tail neutral point.

$$\alpha_t = \alpha + i_t - \epsilon_t + \alpha_{dyn} \tag{7}$$

The downwash angle ϵ_t is modeled proportional to the wing angle of attack: $\epsilon_t = \frac{\partial \epsilon_t}{\partial \alpha}\alpha$. The 2-point aerodynamic formulation considers the downwash lag from the neutral point of the wing to the neutral point of the tail: $\Delta t = r_t^*/V$. As stated above, the ground effect influences the downwash angle at the tail considerably, which can be modeled by an additional downwash angle term multiplied by the ground effect influence function:

$$\epsilon_t = \left[\frac{\partial \epsilon_t}{\partial \alpha} + \left(\frac{\partial \epsilon_t}{\partial \alpha}\right)_{ge} \sigma_2\right] \alpha(t - \Delta t) \tag{8}$$

It should be noted that taking into account the lag effect in the ground effect downwash term the pitching moment is also a function of the aircraft's sink rate in combination with the aircraft's speed. The sink rate dependency was observed for some aircraft and resulted in additional modeling effort, see e. g. [14]. The formulation acc. to eq. (8) was found to cover all evaluated airspeeds and sink rates sufficiently.

Total lift of the aircraft is the sum of wing lift and horizontal tail lift:

$$C_L = C_{L,wb} + \frac{S_t}{S} C_{L,t} \tag{9}$$

The aircraft drag is, acc. to eq. (1):

$$C_D = C_{D0} + (1 - \sigma_1)\frac{1}{e\pi\Lambda} C_L^2 \tag{10}$$

Zero pitching moment is extended by a ground effect term $C_{m0,ge}\sigma_1$. The total pitching moment is simply the sum of zero pitching moment and horizontal tail lift multiplied by the tail lever arm.

It should be noted that two ground effect influence functions are necessary: one to describe lift/drag and zero pitching moment change in ground effect (σ_1), the other for downwash angle change in ground effect (σ_2). Summarizing, the (longitudinal) model only contains 5 parameters for ground effect modeling.

Identification Results

Flight test data from level fly-bys, stationary ones and ones with dynamic elevator inputs, and several landing approaches were evaluated to determine the ground effect model parameters. The parameters were identified for different flap positions separately. *Table 1* lists the identified ground effect model parameters. *Fig. 7* compares both identified ground effect influence functions σ_1, σ_2 for different flap positions.

flap	clean	5°	14°
a (σ_1)	6.9	4.4	5.0
a (σ_2)	3.7	3.8	3.7
$C_{L\alpha,ge}$	1.3	2.3	1.1
$\left(\frac{\partial \epsilon_t}{\partial \alpha}\right)_{ge}$	-0.43	-0.69	-1.03
$C_{m0,ge}$	0.11	0.085	0.111

Table 1: Identified ground effect model parameters for different flap settings

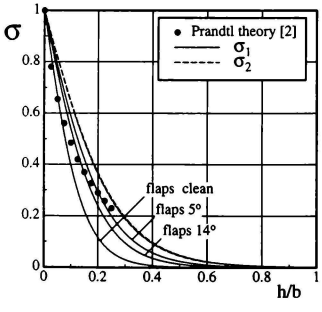


Figure 7: Identified ground effect influence functions σ_1 and σ_2 for different flap settings compared to values given by Prandtl [2]

The downwash angle reduction was found to be more important in ground effect than lift and drag changes. The shape of downwash reduction is independent of flap setting, only the magnitude of the identified $\left(\frac{\partial \epsilon_t}{\partial \alpha}\right)_{ge}$ is rising with increasing flap position, see *Fig. 8.* A

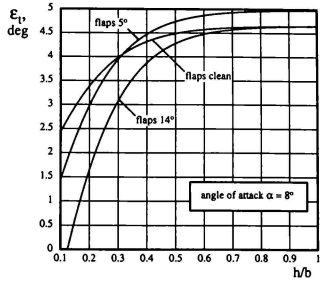


Figure 8: Identified influence of ground effect on downwash angle ϵ_t at horizontal tail for different flap settings

second, but less important term for pitching moment change in ground effect is the zero coefficient $C_{m0,ge}$. This parameter was determined to be positive.

The lift and drag influence function σ_1 *is* flap dependend, but shows no trend with flap setting. This can also be seen in *Fig. 9* for the L/D-increase in ground effect. The theoretical values for a wing with elliptic

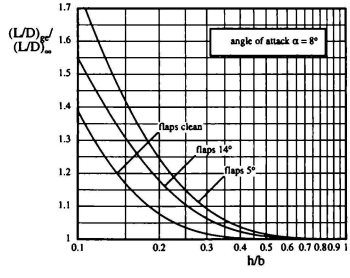


Figure 9: Identified L/D increase in ground effect for different flap settings

lift distribution and a quadratic polare can be computed acc. to eq. (3) with σ-values given by *Prandtl* [2]. For an altitude of a tenth of the wing span, eq. (3) gives an L/D-ratio of $\sqrt{2}$, the identified values are 1.55 (14° flap position), 1.38 (clean configuration), and 1.75 (5° flap position). The identification results seem to

be quite comparable to the theoretical values, only the L/D-increase for 5° flap position seems to be too high.

Summarizing, the main effects in ground effect for the VFW-614 are:

- increase of the lift curve *slope*, no change in the zero lift coefficient
- reduction of the induced drag as theoretically predicted
- reduction of the downwash angle at the horizontal tail, which results in a considerable pitch down moment,
- and a slightly increasing zero pitching moment coefficient.

The effects are summarized in *Fig. 10* for the 14°-flap position, where the total longitudinal coefficients are

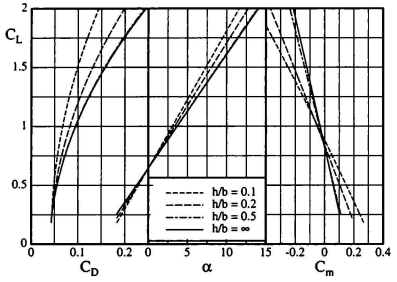


Figure 10: Identified change of aerodynamic coefficients in ground effect for 14° flap settings (flow separation neglected)

computed for different altitudes above ground. Now it can be seen clearly that the pitch down from downwash angle reduction is the dominating effect within the pitching moment coefficient.

Validation

A lot of simulation work was performed in order to validate the identified ground effect model. All recorded level fly-bys were simulated without controller feedback (*Fig. 9*), but allowing a constant control offset in elevator, aileron, rudder, and thrust. The mean radar altitude measurement had to be matched exactly. The resulting control offsets were all within the Level D (II) tolerances [1].

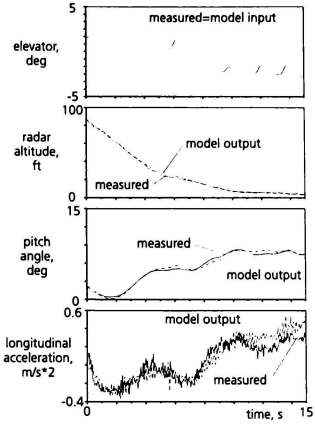


Figure 11: VFW-614 landing approach and flare *with* complete ground effect model, *no* controller feedback

More interesting for validation is the matching of landing approach and flare. *Fig. 11* shows the 14° flap approach discussed above, now using the identified ground effect model for simulation. *No* controller feedback was used: the model control inputs (elevator, thrust) are exactly the measured ones.

It can be seen that pitch angle and altitude matches are well within the Level D quality tolerances, they are nearly perfect. Only the match of the longitudinal acceleration reveals slightly too strong reduction of the induced drag within the model. For clarification, *Fig.*

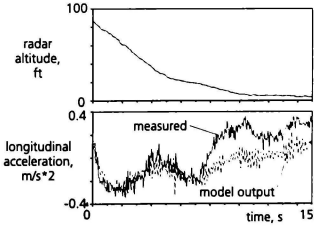


Figure 12: VFW-614 landing approach and flare *with* ground effect model, but *without* reduction of induced drag

12 shows the match for the same landing approach, but neglecting the induced drag reduction in the model by setting $\sigma_1 = 0$ in eq. (10). The amount of difference between measured and modeled longitudinal acceleration is the real induced drag reduction in ground effect.

Conclusions

Flight test data of the twin turbofan aircraft VFW-614 were evaluated with maximum likehood parameter estimation techniques for determination of an aerodynamic ground effect model. The model is formulated as a 2-point model, which describes the lift of the wing and the horizontal tail separately.

The dependency of the aerodynamic coefficients on the altitude above ground is formulated with an analytical approximation of the ground effect influence function stated first by *Prandtl* in 1923. The simple, but very practical approximation provides only one parameter, that was identified for the test aircraft for different flap settings.

The quantity of ground effect influence as a function of altitude was separated out of the flight test data with inverse simulation techniques. This technique gives a basis for model structure determination, as the simulation deficiency (in this case the total incremental model) can be clearly interpreted. The following main aerodynamic changes in ground effect were identified: (a) reduction of the wing downwash angle, which results in a pitch down moment of the total aircraft, (b) increase of the wing lift curve slope, and (c) reduction of the induced drag. The downwash angle of the wing is modeled to be time delayed at the horizontal tail (2-point formulation), thus rendering the model dependent on sink rate.

Comparing the results to theoretical values, the physical plausibility of the identified aerodynamic characteristics was shown, for example, the L/D-increase in ground effect and the shape of the ground effect influence function. For validation a landing approach and flare simulation was compared to flight test data.

References

1. FAA *Airplane Simulator Qualification.* Advisory Circular AC 120-40 C, Edition 1/26/95 (1995)
2. L. Prandtl *Der induzierte Widerstand von Mehrdeckern.* Ergebnisse der Aerodynamischen Versuchsanstalt zu Göttingen, II. Lieferung (1923)
3. C. Wieselsberger *Der Einfluss der Erdbodennähe auf den Flügelwiderstand.* Ergebnisse der Aerodynamischen Versuchsanstalt zu Göttingen, II. Lieferung (1923), S. 41
4. A. Lippisch *Der Aerodynamische Bodeneffekt und die Entwicklung des Flugflächen-(Aerofoil-)Bootes.* Zeitschr. Luftfahrttechnik-Raumfahrttechnik Nr. 10 (1964)
5. A. W. Carter *Effect of ground proximity on the aerodynamic characteristics of Aspect Ratio 1 airfoils with and without end plates.* NASA TN D-970 (1961)
6. R. V. Jategaonkar, E. Plaetschke *Maximum Likelihood Parameter Estimation from Flight Test Data for General Non-Linear Systems.* DLR-FB 83-14 (1983)
7. P. G. Hamel *Aerospace vehicle modelling requirements for high bandwidth flight control.* From: "Aerospace Vehicle Dynamics and Control" by M.V. Cook and M. J. Rycraft, Clarendon Press, Oxford, pp. 1-31 (1994)
8. W. Mönnich *Ein 2-Punkt-Aerodynamikmodell für die Identifizierung.* Symposium 'Systemidentifizierung in der Fahrzeugdynamik', DFVLR-Mitt. 87-22, Paper Nr. 3.1 (1987)
9. R. V. Jategaonkar *Identification of the Aerodynamic Model of the DLR Research Aircraft ATTAS from Flight Test Data.* DLR-FB 90-40 (1990)
10. F. C. Hall *Flight Testing for Simulator Data.* Conference Proceedings AIAA-91-2930-CP, New Orleans (1991)
11. M. P. Fink, J. L. Lastinger *Aerodynamic characteristics of Low Aspect Ratio Wings in Close Proximity to the Ground.* NASA TN D-926 (1961)
12. P. E. Kumar *An Experimental Investigation into the Aerodynamic Characteristics of a Wing, with and without End Plates, in Ground Effect.* College of Aeronautics, Cranfield, Rep. No. 201 (1968)
13. W. v. Grünhagen *Inverse Simulation: a Tool for the Validation of Simulation Programs - First Results.* Zeitschrift für Flugwissenschaften und Weltraumforschung (ZFW), Vol. 17, No.3 (1993)
14. S. Corda et al. *Dynamic Ground Effect Flight Test of an F-15 Aircraft.* NASA Technical Memorandum 4604 (1994)

NONLINEAR DYNAMIC SIMULATION OF A TETHERED AEROSTAT: A FIDELITY STUDY.

S. P. Jones* and L. D. Schroeder†

TCOM, L. P., Columbia, MD

Abstract

Full-scale flight tests of an instrumented TCOM 71M™ tethered aerostat, conducted by the U.S. Army under the JLENS program, provide a data base for determining the fidelity of the TCOM nonlinear dynamic simulation (NLDS). Six Global Position Sensors, accurate to ± 5 cm, determined the position and attitude of the aerostat and winds were measured using a 3-axis anemometer mounted on the aerostat's fin near the tip. A 32-minute window of data was selected for comparison with the NLDS using the recorded winds to derive a turbulence table for the simulation. This required correcting the winds for aerostat velocity and transforming to an Earth-fixed coordinate system aligned with the mean aerostat heading. Graphical comparisons are made in the time domain of the aerostat's position and orientation as well as tether tension. Qualitatively, the comparisons are very good, generally showing the same patterns of motion and order of magnitude of the standard deviations. Tether tension in particular showed a remarkable duplication by the simulation. Frequency spectra based on Fast Fourier Transformation of the time histories are also compared.

Introduction

In March of 1997, the U. S. Army, under the JLENS‡ program, conducted flight tests of a highly instrumented TCOM 71M™ tethered aerostat at the McGregor Range of the White Sands Missile Range. These tests provided data on the motions of the aerostat in various levels of turbulence for payload design and to assess the fidelity of flight simulation computer programs. This paper deals with a comparison of selected flight data with the motions and tether tensions predicted by the TCOM Nonlinear Dynamic Simulation (NLDS) using the measured, time-dependent wind vector as an input. The comparison is made in the time domain, comparing the position, orientation and tether tension as a function of time. This is a much more stringent test of fidelity than comparisons made on a statistical basis or those relying solely on the frequency spectrum of the motions. It is a test not only of the flight dynamic model, but also of the "frozen-field" wind model which is assumed by the simulation.

The TCOM NLDS has been in development for well over 20 years. The first version was published in 1982[1] and it has been refined and re-coded in modern computer language with many options and improvements added in the intervening years, including the full implementation of the aerodynamic model of Jones and DeLaurier.[2] It has been the tool used in a number of studies, including the dynamics of the STARS aerostat[3] and the simulation of a moored aerostat.[4] The NLDS was used in the design and development of the 71M™ aerostat, which was "flown" with the simulation to determine its stability and response to turbulence before the first gore was cut. Studies with the NLDS have been used to determine the optimum free lift for safe aerostat operation and to provide guidance for the design strength of tethers. It has been an important tool in accident investigation by simulating the flight involving an incident. Thus, verification of its veracity is of utmost importance.

The NLDS is a six-degree-of-freedom dynamic model with a finite-element tether. In the aerodynamic model the hull is divided into a number of panels or longitudinal segments which permit a nonuniform flow as turbulent gusts are propagated from nose to tail. This variation in local angle of incidence produces moments due both to turbulence and rotation. A similar model has been developed by Evans and DeLaurier for powered airships.[5] The force on aerostat hulls resulting from turbulence has been controversial. Based on Calligeros and McDavitt,[6] the present model does not include forces due to fluid acceleration, since the turbulence is

*Consulting Scientist. Senior member AIAA.
†Programmer Analyst.
‡Joint Land Attack Cruise Missile Defense Elevated Netted Sensor System.

Copyright ©1999 The American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

represented as a "frozen-field" or traveling wave translated at the mean wind speed. Tischler and Jex[7] not only include these forces, but assert the existence of an additional force due to a presumed pressure gradient associated with the fluid acceleration. Etkin has taken issue, particularly with the latter point.[8, 9] Etkin and D'Eleuterio[10] have objected to the inclusion of convective acceleration terms in the present model as derived by Jones and DeLaurier. In response, the authors have shown that these terms are required by classical theory.[11]

Of equal importance is the wind input model. Unlike airplanes and airships which create their own relative wind, usually large compared with the ambient wind, the tethered aerostat is anchored in one location, subjected to the ambient mean wind and turbulence, with its motions constrained by the tether. Thus the "frozen-field" model is not the same. In the former, the turbulence is assumed to be frozen in space and the vehicle is translated through it,[12] while for the tethered aerostat the turbulence is modeled as a frozen volume translated through the vehicle at the mean wind speed. Verification of this model is an important aspect of these studies.

The opportunity to verify a simulation with real flight data is rare, primarily because of the expense involved and the need for restricted air space. During the late 1970's, TCOM instituted a program of instrumented flight of a 365 aerostat on Grand Bahama Island, with the objective of obtaining data to verify the NLDS. Those results, published in Ref. 1, were thought to duplicate the recorded flight data fairly well; however, some of the parameters, particularly the tether tension, were greater than actual. Since then the code has been improved in many ways. The present studies are therefore important in reexamining the fidelity of the simulation with improved technology for measuring and recording data. They have already lead, in the course of this study, to an improvement in the code with respect to the wind model and its interface with the aerodynamic model. These flight data will continue to provide a base for testing new developments and modifications of NLDS codes.

Sources of error in these studies include deviations of the real winds from the idealized turbulence model. The wind measurements are subject to question due to proximity of the 3-axis anemometer to the aerostat. In addition, the boundary conditions differ for the flight data and the simulation. For the latter, the aerostat was initially in static equilibrium, while the real aerostat was in dynamic motion. For integrations, as in the simulation, errors may accumulate. Thus, an exact duplication by the NLDS should not be expected.

Coordinate Systems and Nomenclature

In both the simulation and the full-scale flight data, the aerostat's position and orientation are described relative to an Earth-fixed coordinate system (EFS), depicted in Fig. 1. The winds, which were measured with a 3-axis anemometer mounted on the port fin, were recorded relative to a body-fixed system (BFS). The two systems are related by a transformation matrix which rotates the latter through roll, pitch and heading angles and by the vector location of the BFS relative to the EFS as shown in Fig. 1. Forces and moments are computed in the former and integrated in the latter. In the NLDS, the EFS is fixed by the initial position of the aerostat in static equilibrium, with x parallel to the mean wind direction. Likewise, in the body-fixed system, x' is parallel to the aerostat's longitudinal axis. The coordinate systems are conventional with x, y and z being forward, to starboard and down, respectively. The rotations, roll, pitch and heading, are clockwise about the x, y and z axes, respectively. The components of the

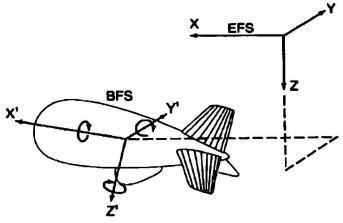


Figure 1 Earth-fixed and body-fixed coordinate systems of the NLDS.

wind vector parallel to x, y and z are u, v and w, respectively, with the subscript, g, indicating the turbulence component, which is the standard deviation of the fluctuations in velocity.

The Aerostat and Instrumentation

The geometric and other parameters of the TCOM 71M™ aerostat are given in Table 1. It has an inverted "Y" empennage and can carry an 1,800 kg payload to 4,500 m and, due to a power tether, stay at that altitude

for a number of days. A Kevlar™ reinforced tether provides power to the system and communications through a fiber optic link.

Table 1
Parameters of the 71M™ Aerostat

Aerostat length	71 m
Aerostat volume	16,658 m^3
Fineness ratio	3.3
Gross weight	4,377 kg
Helium lift	8,073 kg
Tether unit weight	0.647 kg/m
Tether diameter	2.423 cm

The instrumentation for determining aerostat position and orientation consisted of six Global Position System (GPS) sensors capable of measuring position in Earth-fixed coordinates with an accuracy of ± 5 cm. They were located at the nose, near the tail, forward port and starboard, and aft port and starboard. The data from these sensors were used to compute, in one-second intervals, the x, y and z location of the aerostat in EFS coordinates and its pitch, roll and heading. Velocities were computed by integrating the position and orientation and used to correct the anemometer data for aerostat motion. The tether tension was sensed at the confluence point and at the winch on the ground.

There was only one sensor for measuring the instantaneous wind vector, a Gill type 3-axis anemometer located on the port fin near the tip. Unfortunately, it was located very close to the fin, the lower vane being only 60 cm from the surface. Thus, the flow was undoubtedly accelerated and the wind measurements probably higher than in the free stream, although this effect is impossible to quantify. The wind measurements were corrected for position and transformed to true relative wind components in the BFS. To be meaningful, these winds were corrected for the aerostat's translational velocity obtained by integration of the position. Finally, to be useful in this analysis, the winds were transformed to true winds in the EFS of the measurements.

NLDS Wind Model

The method of applying winds to the NLDS is patterned after that described in MIL-F-8785B with modifications for a tethered aerostat. The mean wind is assumed to be constant, parallel to the x-axis of the EFS, which is fixed by the initial position of the aerostat in static equilibrium. The turbulence is the variation in velocity in each of the coordinate axes and in the standard model is isotropic above 533 m.[17] Of course, in the real world, this may not be the case. In normal use of the NLDS, the turbulence is synthesized using a Dryden power spectrum as described in Refs. 1 and 4. The turbulence, or fluctuations in velocity, is translated through the theoretical model at the mean wind speed. For powered vehicles such as airplanes and airships, this mean translation velocity is the vehicle's speed, but for a tethered aerostat it is the ambient wind velocity. This is an adaptation of the "frozen-field" model in which the turbulence in each coordinate axis appears to be frozen in space and translated at the mean wind speed.

For comparing the NLDS with real flight, a window of data should be selected which is at a constant altitude and has a reasonably constant wind direction, which can be judged by the wind vector or the aerostat's heading angle. In order to conform to the model and make comparisons possible, the EFS in which the measurements were made must be rotated about the z-axis so that the x-axis is parallel with the mean wind direction.

To apply the real winds to the NLDS model, a wind table is made containing the real wind fluctuations about the mean in each of the orthogonal directions in the rotated EFS. Ideally, the mean wind parallel to the y and z axes should be zero and that in the x direction should be the velocity of the frozen field or the propagating velocity. Since the wind table represents turbulence or fluctuations in velocity, the mean must be subtracted from each component of velocity in the wind table. In summary, the following steps are taken in preparing a wind table from the flight data.

- Select from the data a window of significant time interval, in which the wind direction is relatively constant as judged by the aerostat's heading.
- Transform all the data to an Earth-fixed coordinate system with the x-axis parallel to the mean heading angle.
- Make a wind table by subtracting the mean value from the u, v and w components of the wind. The mean value of the u wind is the wind speed, which is assumed to be the propagation velocity.

Flight Data Selected for Simulation

A window of data was selected from Flight 48 with wind components shown in Fig. 2. The 1900 seconds (32 minutes) of real wind data have been transformed to an EFS rotated 264.34° from that in which the data were recorded in order that the u component is parallel to the

mean aerostat heading. The mean values and standard deviations are given in Table 2. Note that the u wind has a steady component of 10.3 m/s which represents a mean wind speed of 20 knots.

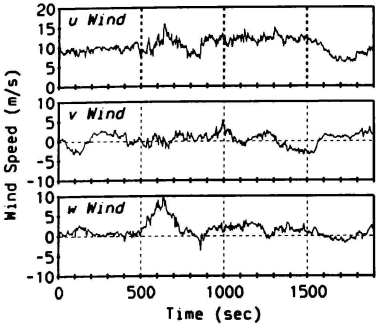


Figure 2 Wind data from Flight 48 from 20:58:13 to 21:29:53 GMT. The Earth-fixed coordinate system has been rotated 264.34° for application to the NLDS.

Table 2
Mean and Standard Deviation of Winds in Selected Window of Data

Vector Component	Mean (m/s)	S. D. (m/s)
u (longitudinal)	10.30	1.898
v (lateral)	0.39	1.606
w (vertical)	1.46	1.908

A significant feature of the winds, shown in Fig. 2, is a large updraft beginning at about 500 seconds, in which the peak upward velocity reaches 10 m/s or 19 knots.

There is a substantial mean value to the vertical component of wind due to the large updraft and possibly the accelerated and diverted flow on the anemometer due to the proximity of the fin. In order to make a table of turbulence applicable to the NLDS, the mean values of the winds must be subtracted as stated above. This obviously introduces some error in representation of the winds. Another source of error is the assumption that the mean wind is constant and in the same direction. This introduces a certain ambiguity between the u and v components of the turbulence. If the real mean wind varies in speed or direction it will be interpreted by this model as turbulence and may exaggerate the lateral turbulence which will contain a component of the mean, propagating wind. Such an ambiguity will not exist for the vertical component of turbulence.

NLDS Inputs and Outputs

The NLDS is one of a family of supporting programs. The input data file is generated by a static analysis program which contains the weight and balance and other data on the aerostat as well as environmental parameters. Some of the input data for this simulation are given in Table 3.

Table 3
Flight Data and Static Inputs for the Simulation

Flight altitude	2587 m
Pad altitude	1354 m
Pad temperature	22°C
Pad pressure	862 mb
Helium lift	8074 kg
Ballonet air volume	6289 m^3
Mean wind speed	20 kn
Static pitch angle	7.2°
Static tether tension	3938 kg

Another program reads the NLDS output and presents it in statistical or graphical form. This output usually consists of 34 parameters dumped to the hard drive at one-second intervals throughout the run. The data dump time can be varied down to the time interval of the integration which, in the present case, was 0.05 sec. The number and type of parameters computed and output can also be varied. Normally included are the aerostat's location in EFS coordinates, angular attitude in pitch, roll and heading, vector components of the translation and rotational velocities and tether tensions at the aerostat and at the winch, as well as accelerations at the payload or any other location on the aerostat.

Results of the Simulation in the Time Domain

A simulation run was made with a wind table derived from the data in Fig. 2. The results are shown in Figs. 3 through 9, in which the NLDS output is compared with the actual flight data. Figure 3 shows that the NLDS reproduces the tether tension of the aerostat over the 32-minute period with remarkable fidelity, although the standard deviation is slightly larger. This may be attributed to the accelerated flow over the anemometer giving an apparent wind velocity greater than that in the

free stream. This may also account for the exaggerated downward pitch in response to the updraft at about 650 seconds in Fig. 7. This greater downward pitch is responsible for the charge forward at the same time (Fig. 4). The downward pitch and charge forward in response to updrafts is a common phenomenon in desert regions such as White Sands. For other parameters, the simulation appears to follow the flight data to a reasonable degree, considering the potential sources of error.

To assess the statistical response to turbulence, another NLDS run was made using the winds of Fig.2 in the 1000-sec. period from 900 to 1900 sec. This excludes the updraft, which is considered a discrete event and not representative of normal turbulence. The results in graphical form are essentially the same as for the longer simulation, but the statistical data, presented in Table 4, exclude the singular updraft. The agreement in x, y and z location and in heading is good, all being within 20% of the standard deviations for the flight data. Tether tension and pitch, although duplicated by the NLDS to a remarkable degree, are significantly higher as has already been noted. The greatest percentage deviation is seen in roll, although the magnitude of the roll, real and simulated, is less than five degrees.

Table 4
Standard Deviations of Parameters from NLDS Compared with Flight Data

Parameter	Flight Data	NLDS
x Location (m)	31.6	30.7
y Location (m)	55.4	63.1
z Location (m)	3.1	2.6
Pitch (deg)	1.2	1.9
Roll (deg)	1.2	2.1
Heading (deg)	12.8	11.1
Tether tension (kg)	353	497
u_g (m/s)	2.2	2.2
v_g (m/s)	1.7	1.7
w_g (m/s)	1.3	1.3

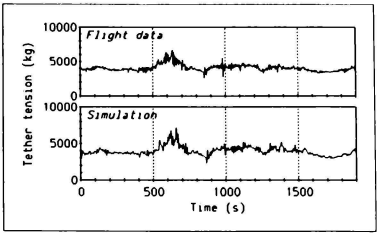


Figure 3 Tether tension from flight data and NLDS.

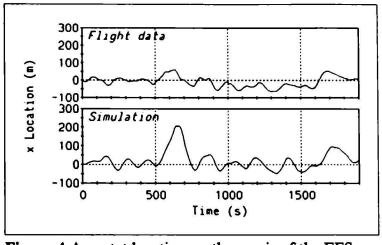


Figure 4 Aerostat location on the x axis of the EFS.

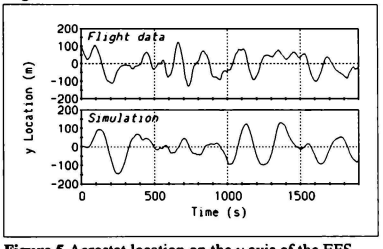


Figure 5 Aerostat location on the y axis of the EFS.

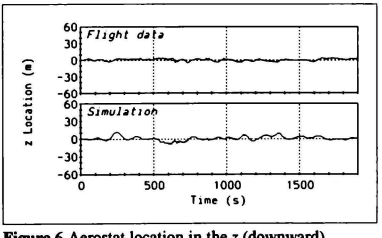


Figure 6 Aerostat location in the z (downward) direction.

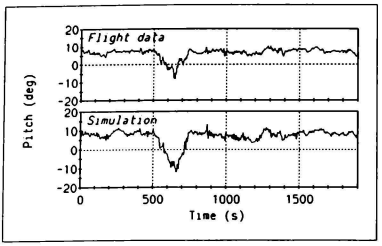


Figure 7 Aerostat pitch angle from flight data and NLDS

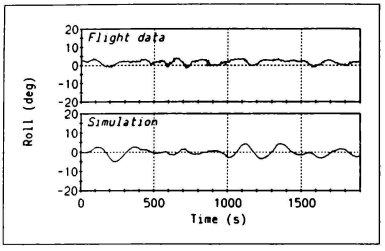


Figure 8 Aerostat roll angle from flight data and NLDS.

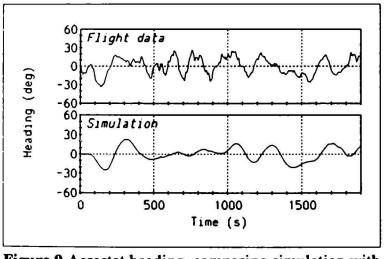


Figure 9 Aerostat heading, comparing simulation with flight data.

Frequency Spectra

Another important aspect of the simulation's fidelity is in the frequency spectra of the various parameters. This was determined by Fast Fourier Transformation (FFT) of the time-history data, which gives a frequency distribution dependent upon the number of data points used in the analysis. The spectra of two important parameters, tether tension and lateral displacement are presented in Figs. 10 and 11, respectively, where frequency distributions derived from the simulation and the flight data are compared. The comparisons, which are necessarily qualitative, show a correspondence in the dominant frequencies. The plots also show that the energy is concentrated at very low frequencies, with very little above 0.1 Hz.

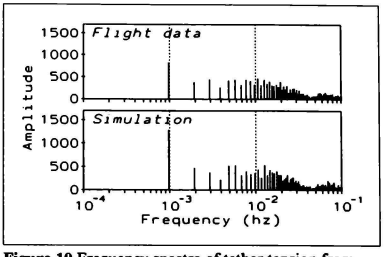


Figure 10 Frequency spectra of tether tension from flight data and the NLDS.

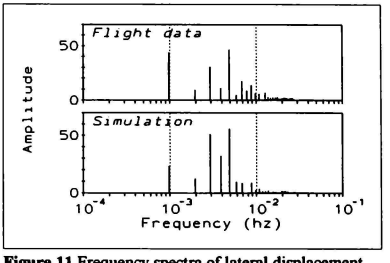


Figure 11 Frequency spectra of lateral displacement from flight data and the NLDS.

Conclusions

This study demonstrates that real flight data of a tethered aerostat, including recorded relative winds, can be used to assess the fidelity of a dynamic simulation computer program. This requires recording the data in an Earth-fixed coordinate system which can be transformed to coincide with the mean wind axis as required by the simulation. The wind model, in which the mean wind is assumed to be of constant speed and direction with the perturbations superimposed in each of the coordinate axes, is a satisfactory representation with

a properly selected window of data. In this model the frozen-field concept is applied to an ambient wind which propagates the turbulence.

Much of the evaluation of fidelity by comparing the NLDS output to the flight data is qualitative. In that respect, the response of the simulation to the vertical component of the real wind is excellent in reproducing the tether tension and pitch angle of the flight data in detail, including the response to a strong updraft. That the magnitude of the simulated tension and pitch is somewhat exaggerated can probably be accounted for by the accelerated flow on the anemometer, indicating a higher velocity than actually existed. Accelerated flow near the fin would also be enhanced by an increased angle of attack due to the updraft. The vertical motion in the simulation is greater than recorded by the aerostat (Fig. 6) but both are less than 12 m displacement from the initial position. The longitudinal location (Fig. 4) is reproduced very well, except for response to the updraft, which is also exaggerated. As noted, the pitch down and charge forward is typical behavior of tethered aerostats responding to updrafts commonly seen in desert regions.

The lateral motions, y location, heading and roll, generally follow the flight data but in less detail than in the vertical and longitudinal motions. These are likely more sensitive to deviations of the real winds from the idealized frozen-field wind model. The aerostat heading (Fig. 9) shows a relatively high frequency component not found in the simulation, the origin of which is unknown at this time.

The statistical response to turbulence, as shown in Table 4, is satisfactory for this type of analysis. Here also tether tension and pitch show greater standard deviations in the simulation. The frequency distributions obtained by FFT of the time histories of tether tension show that the simulation contains the higher frequencies due to tether oscillation seen in the flight data (Fig. 10). In Fig. 11, these higher frequencies are absent or insignificant in the lateral motion.

The simulation is sufficiently close to the real flight data to validate the aerodynamic model, which excludes the effects of fluid acceleration as proposed in Ref. 7 and includes forces due to convective accelerations as discussed in Refs. 10 and 11.

The weakness of these flight tests is in the quality of the wind data, which suffers from the unfortunate placement of the 3-axis anemometer in the boundary layer of the fin. This should be corrected in future experiments, possibly by placing the anemometer at the aerostat's confluence point. Even with this deficiency, however, the wind data is remarkably useful and illustrates the importance of on-board wind measurements from which detailed time-history comparisons can be made.

References

[1]Jones, S. P. and Krausman, J. A., "Nonlinear Dynamic Simulation of a Tethered Aerostat," *Journal of Aircraft*, Vol. 19, August 1982, pp 679-686.

[2]Jones, S. P. and DeLaurier, J. D., "Aerodynamic Estimation Techniques for Aerostats and Airships," *Journal of Aircraft*, Vol. 20, February 1983, pp 120-126.

[3]Jones, S.P., Krausman, J.A. and Sunkara, B.D., "Dynamics of the STARS Aerostat," AIAA-83-1990, AIAA Lighter-Than-Air Systems Conference, Anaheim, California, July 1983.

[4]Jones, S. P., "Nonlinear Dynamic Simulation of a Moored Aerostat," AIAA-87-2505, AIAA 7th Lighter-Than-Air Technology Conference, Monterey, California, August 1987.

[5]Evans, J. R. and DeLaurier, J. D., "The Shenandoah Flies Again: A computer Simulation," AIAA 81-1325, July 1981.

[6]Calligeros, J. M. and McDavitt, P. W., "Response and Loads on Airships due to Discrete and Random Gusts," Bureau of Aeronautics, Department of the Navy, Contract No. 56-825-6, MIT Technical Report 72-1, Feb. 1958.

[7]Tischler, M. B. and Jex, H. R., "Effects of Atmospheric Turbulence on a Quadrotor Heavy-Lift Airship," *Journal of Aircraft*. Vol. 20, December 1983, pp. 1050-1057.

[8]Etkin, B., "Comment on 'Effects of Atmospheric Turbulence on A Quadrotor Heavy-Lift Airship," *Journal of Aircraft*, Vol. 22, January 1985, pp. 93-95.

[9]Tischler, M. B. and Jex, H. R., "Reply by Authors to B. Etkin," *Journal of Aircraft*, Vol. 22, January 1985, pp. 95-96.

[10]Etkin, B. and D'Eleuterio, G., "Comment on 'Aerodynamic Estimation Techniques for Aerostats and Airships," *Journal of Aircraft*, Vol. 22, November 1985, p.1023.

[11]Jones, S. P. and DeLaurier, J. D., "Reply by Authors to B. Etkin and G. D'Eleuterio," *Journal of Aircraft*, Vol. 22, November 1985, pp. 1023-1024.

[12]Chalk, C.R., Neal, T.P., Harris, T.M. and Pritchard, F.E. "Military Specifications--Flying Qualities of Piloted Airplanes," *Background Information and User Guide for MIL-F-8785B(ASG)*, Air Force Dynamics Laboratory, Air Force Systems Command, Wright-Patterson AFB, Ohio; see also AFFDL-TR-69-72, Aug. 1969.

AIAA-99-4326

HYPERVELOCITY VEHICLE MODELING AND SIMULATION USING GENESIS

Phillip D. McKeehen*

Air Force Research Laboratory(AFRL/VAAD)

Wright-Patterson Air Force Base, Ohio 45433-7521

ABSTRACT

This paper discusses the use of GENeral Environment for the Simulation of Integrated Systems(GENESIS) in the modeling and non-realtime 6 DOF simulation of a Two Stage To Orbit (TSTO) hypervelocity vehicle which uses a combination of scramjet and rocket propulsion modes. The vehicle configuration is the highly swept double-delta planform defined in the 1991 Northrop-Grumman Modern Aerospace Vehicles Robust H-Infinity Control (MAVRIC) contract with the USAF. The discussion topics include: an overview of the Fortran 77-based GENESIS software's structure and basic functionality; the simulation's aerodynamic, propulsion, and structural models; linear model generation and control law design issues. Examples of simulation output time-histories for simple maneuvers are shown.

INTRODUCTION

A major area of current research in the aerospace community is concerned with the analysis and design of transatmospheric hypervelocity vehicles. These include Single Stage To Orbit (SSTO) and Two Stage To Orbit (TSTO) configurations, equipped with combinations of air-breathing and rocket propulsion systems. In conjunction with the recent expansion of national interest in these vehicles, the Air Force Research Laboratory's Flight Dynamics and Control Branch is also increasing its research in this technical area.

The concept of using hypersonic vehicles as aerospace planes in missions such as delivering payload to earth orbit, military reconnaissance and high speed cruise was studied in the 1980s' X-30 and NASP programs. These vehicles used a SSTO method for achieving low Earth orbit, requiring several types of propulsion systems to be activated during different flight phases (subsonic , transonic, supersonic, hypersonic and orbital).[1] Other researchers have studied TSTO configurations for similar missions. In addition to the necessity for multi-phase propulsion systems, the TSTO share with the SSTO configurations the technical complications of a high degree of coupling between aerodynamic, propulsive, inertial and structural forces and moments on the vehicle. In addition, research has shown that surface temperature variation can be extreme enough during different flight phases to alter material stiffness and lower fuselage bending modal frequency.[1]

The development of high quality control laws for such vehicles should be based on high-fidelity models and simulations which account for all of the pertinent complicating factors mentioned above. Thus, during the past year the Flight Dynamics and Control Branch has expended much effort in the development and validation of corresponding high fidelity mathematical and software models and their associated integration into the GENESIS nonlinear 6DOF simulation software environment.

Initial work focussed on a parallel path of model and simulation development for two different hypervelocity vehicles of interest.These models and their potential upgrades to include all of the complicating coupling effects may be used for analysis and design of control laws and vehicle flying qualities analysis in addition to their use as a means of investigating different modeling techniques for the aerodynamics, structural effects, thermal effects, propulsion systems and their coupling.

* Senior Member AIAA

This paper is declared a work of the U.S.government and is not subject to copyright protection in the United States.

An SSTO horizontal take-off and landing vehicle model was obtained from NASA and initial work was done to implement the model in GENESIS. This model includes components for a multi-mode air-breathing engine and an auxilliary rocket propulsion system. The model is based on a generic lifting body configuration which uses the fore part of the underside of the vehicle to compress and feed air to the engine and the aft part to expand the flow out of the engine. Specific model components include aerodynamics, propulsion (and its interaction with aerodynamics), and mass / inertias. Control during atmospheric flight is achieved using all movable wings and dual vertical stabilizers with rudders.

Following the initial development of a GENESIS-based aerodynamics model component for this SSTO vehicle simulation, the decision was made to prioritize the development of a TSTO vehicle model and simulation. The SSTO model and simulation is therefore incomplete. The TSTO vehicle chosen is the Northrop–Grumman Modern Aerospace Vehicles Robust H-Infinity Control (MAVRIC) vehicle , studied in the corresponding contract from 1987-91[2,3]. The primary justification for prioritizing this vehicle simulation is the fact that a complete GENESIS-based Fortran simulation of it was already delivered in GENESIS V1.97 for the VAX/VMS computer. A major task this year has been to re-host and re-validate the simulation software on the SGI/UNIX system under the updated and greatly modified GENESIS V2.2. With Northrop-Grumman providing key technical support, this critical task was completed in April 1999.

The MAVRIC configuration uses a highly-swept double-delta planform with semispan-to-length ratio of approximately one-fourth (Fig 1). The aerodynamic controls include elevons, which can be used symmetrically or differentially, a rudder and a body flap. The three propulsion modes include scramjet, rockets and reaction control rockets and allow a mission starting from horizontal take-off and eventually achieving low Earth orbit. Note that only the second stage (Mach = 10 to orbit) was modeled and simulated in the original software delivered in 1991. A recent task has been the expansion of the MAVRIC aerodynamic model to include Mach numbers below Mach 10 and the incorporation of this expanded aerodynamic model in the new SGI / UNIX GENESIS V2.2 simulation software.

The remainder of this paper consists of an overview of the GENESIS software's structure and basic functionality; discussion of the MAVRIC simulation's aerodynamic, propulsion and structural models; linear model generation and control law design issues. The paper concludes with examples of simulation output time-histories for simple maneuvers.

THE GENESIS SOFTWARE ENVIRONMENT

GENESIS is a proprietary Fortran 77 software product of the Northrop-Grumman Corporation[4]. It is supported on a number of different platforms: VAX/VMS, IBM/TSO, SUN/UNIX, SGI/UNIX, HP/UNIX, and PC/DOS. GENESIS is capable of performing in either a non-realtime or real-time capacity. Its structure and functionality are described in detail in [4].

The GENESIS software can be used to simulate any time-varying system and has been used to simulate guided weapons, mass transit vehicles and automobiles. In addition to the MAVRIC vehicle simulation described in this paper, aerospace simulation applications at AFRL have included a supermaneuverable F/A-18[2,3], a tailless vehicle with innovative control effectors[6,7,8], the F-15 / SMTD and various configurations of the VISTA/F-16[5]. Although generic, the software evolved from aircraft simulation applications. For this reason, many standard aircraft simulation features are built-in. For example, modules exist for the 6 DOF rigid body aircraft equations of motion, a standard atmosphere model and standard disturbance models. Only such vehicle specific subroutines as actuators, propulsion system, airframe, landing gear and control laws or various databases (e.g. aerodynamics and propulsion) must be developed by the user (Fig 2).

Besides the standard components, GENESIS has simulation core software comprised of built-in routines for linear model extraction, extensive trimming capabilities, database management and many utility functions (control system filters, matrix manipulations, etc.). The extensive and powerful user interface incorporates a wide range of commands that allow the performance of a diverse array of system analyses and a variety of simulation development debugging tasks.

The flexibility and ease of use of the software structure shown in Fig 2 is further enhanced by the

hierarchical use of distinct libraries for configuration management (Fig 3). At the lowest priority level is the simulation library, containing generic simulation subroutines which can be used by any GENESIS simulation (tailless aircraft, VISTA/F-16, F-15/SMTD etc). The next higher level library is the project library. Here one library would be specifically for the tailless aircraft, a separate one for the F-16 and a separate one for the F-15/SMTD. The highest priority library is the user project library. Here a user may have additional routines or slightly modified routines for a specific vehicle simulation which take precedence over the lower level routines.In this way a user may still run the current version of a particular simulation while simultaneously developing and debugging a new version of it and maintaining complete configuration control.

MAVRIC AERODYNAMIC MODEL

The aerodynamic control effectors include elevons, rudder and body flap. Aerodynamic forces and moments (lift, drag, side force, pitching moment, rolling moment, yawing moment) are calculated based on the use of build-up of total coefficients as a sum of basic coefficient plus increments due to control surface deflection plus dynamic derivative terms. The basic coefficients and increments due to surface deflections are stored in a database as nonlinear functions of multiple aircraft state variables, while traditional linear methods are used to estimate the dynamic derivatives. Subroutine HYPAFM.f builds the complete aerodynamic model and contains the code for calls to the GENESIS built-in database extraction routine. The code implements both rocket mode and scramjet mode aerodynamics. As an example, the calculation of pitching moment in either mode is done from the equation :

$$C_m = C_{m_b} + \delta C_m + (C_{m_q} q\bar{c}) / (2v)$$

with $$\delta C_m = \sum_{i=1}^{n} e_i \delta C_{m_i}$$

where:

C_m = total pitching moment coefficient

C_{m_b} = basic pitching moment coefficient of airframe, controls undeflected

δC_m = total pitching moment contribution from control surfaces

n = number of control effectors contributing to δC_m

e_i = individual surface control effectiveness (e_i = 1 indicates same effectiveness as in wind tunnel tests)

δC_{m_i} = pitching moment coefficient of individual surface

C_{m_q} = pitch damping derivative

q = pitch rate

$\bar{c}$ =mean aerodynamic chord

v = free stream velocity

The other five aerodynamic force or moment coefficients are calculated similarly.

The aerodynamic model of the MAVRIC vehicle in the original GENESIS simulation considered the flight regime from staging to orbit only. This model contained aerodynamic data from Mach 10 to 26. The data was obtained from the Supersonic/Hypersonic Arbitrary Body Program[9], (SHABP Mk IV), a finite element analysis code which computes the loads on a vehicle by applying user specified pressure methods to each vehicle component. Over a dozen high-speed pressure methods are available for the user to choose from. The original SHABP analyses used Newtonian theory on all vehicle components. To simulate flight regimes from orbit to landing, aerodynamic data was required for landing up to Mach 10.

For the low speed portion of the flight envelope (Mach=0.2 to 2), Digital Datcom[10] was used to generate aerodynamic coefficients and derivatives. Digital Datcom did not provide roll damping (Clp) predictions for this configuration, so the Slender Body Theory method presented by Nielsen[11] was used. Digital Datcom did not provide any estimates of rudder effectiveness either, so the technique given by Roskam[12] was used.

From Mach=2 to 10, data was generated using the SHABP Mk V or "VECC" code[13]. VECC (Viscous Effects for Complex Configurations) is the latest release of the SHABP code and contains automated control deflection and stability derivative calculations, along with a graphical user interface. Three pressure method combinations were studied to see which gave the most credible results at low supersonic Mach number and blended best with the Digital Datcom results. These combinations were: (1) Newtonian flow on the entire vehicle, (2) Newtonian flow on the windward side and Prandtl Meyer expansion on the leeward side, and (3) Tangent Wedge/Tangent Cone on the windward side and Dahlem-Buck mirror/ACM empirical on the leeward side. A comparison of predicted lift coefficients is shown in Fig 4. At high Mach numbers, all of the VECC pressure method options give essentially the same result. They diverge at the lower Mach numbers, and the Newtonian/Prandtl Meyer expansion combination was found to match best with the Digital Datcom results. This was the method used for the VECC results included in the simulation.

To blend the Digital Datcom and VECC results, linear interpolation was used between the Digital Datcom results at Mach 2 and the VECC results at Mach 4. The pitch rate derivatives (CLq, Cmq) were not provided by Digital Datcom above Mach 1, so the Mach 0.8 results were faired into the VECC results at Mach 2. The pitching moment at zero angle of attack predicted by Digital Datcom was zero between Mach 1 and 2. To provide continuity, a pitching moment increment was added to the Digital Datcom results based on the VECC value at Mach 2.

MAVRIC PROPULSION MODEL

The MAVRIC contract included a detailed assessment of mission requirements followed by an iterative conceptual design process. The TSTO design emerged and a scramjet was included in the configuration since it has much higher effective propulsive efficiency than a rocket for a fairly large Mach number range. This in turn reduces the fuel weight necessary for powered ascent.[2] The MAVRIC vehicle ascent trajectory is :

(1) Takeoff (Wgt = 769K lb)
(2) Separation (M=10 Alt = 120K ft
Wgt = 371.5 K lb)
(3) Scramjet / Rocket Transition
(M= 15 Alt = 150K ft Wgt = 333.4 Klb)
(4) Rocket Shut –Off (M=25 Alt = 400K ft
(Wgt = 153.1 Klb)
(5) From-Orbit maneuvering
(20 deg Synergetic Plane Change)

The scramjet is modeled as a tabular database containing thrust, specific impulse, propulsive lift and propulsive pitching moment as functions of angle of attack, fuel/air equivalence ratio and flight condition. The database region for the scramjet model is from Mach 10 , Altitude 105,000 ft to Mach 15 , Altitude 150,000 ft.

The rocket model is based on technology of the Space Shuttle Main Engine, with the two MAVRIC rockets sized to collectively generate 1 g forward acceleration at Mach 15. It is assumed that fuel flow rate is a linear function of throttle setting and there is constant specific impulse of 450 seconds.

As for the rocket model, the reaction control thrusters modeled are the actual primary and vernier thrusters of the Space Shuttle. Thrust level is 872 lbs per primary jet with a specific impulse of 280 seconds and 25 lb per vernier jet with a specific impulse of 265 seconds. More detailed information concerning the complete propulsion model for the MAVRIC vehicle may be found in [2], from which this basic overview was taken.

MAVRIC STRUCTURAL MODEL

Because of the similarity between the MAVRIC and the National Aerospace Plane (NASP) configurations, and the fact that size and mass of the two vehicles are proportionate, MAVRIC sructural properties were obtained from a NASTRAN analysis of NASP. This analysis determined frequency and bending mode shapes for the first three modes for both longitudinal and lateral-directional axes. Torsional mode shapes were also determined for the first three lateral-directional modes. The flexible model of the MAVRIC is based on second order approximations of the primary bending modes, with the model generating incremental accelerations and rotational rates at the sensor locations due to structural bending and torsion. The increments are then added to the rigid body values at each simulation time step to form the total accelerations and rates. Note that since the primary purpose for developing a model of the MAVRIC in the original contract was to

investigate H^{∞} robust control law design methods, the model only includes the first two longitudinal and lateral-directional modes. The third mode was accounted for as uncertainty in the control law designs. As for the propulsion model, this structural model information is taken from [2], where more detailed information may be found on all other aspects of the vehicle modeling (actuation, atmosphere and atmospheric disturbances , sensors, equations of motion, mass/inertias).

LINEAR MODEL GENERATION

Traditional control design methods are based on linearized models of the vehicle dynamics at discrete operating points (flight conditions) in the flight envelope. A different linear design is done for each flight condition and then the resulting gains are interpolated in an ad hoc manner to provide a full envelope design. Although such control design methods as dynamic inversion do not require sets of linear models, GENESIS has primarily been used with linear design methods. Thus, Volume III from [4] is a highly detailed exposition of how GENESIS can be used to generate corresponding linear models at a given flight condition from a complex nonlinear system model. Recalling that complete GENESIS models are actually built up from several individual models of specific components of the total dynamic system, the following account is a condensed and simplified version of what is presented in detail in [4]. In the context of the linearization discussion below the term *components* actually refers to a grouping of the basic component model functional elements shown in Fig 2.

First, the user must decide exactly which individual functional elements are to be grouped into a component to be linearized. Typical examples of component groupings might be:

(i) Open – Loop Airfame component
(GENESIS aerodynamics element + 6DOF equations of motion)

(ii) Open – Loop Airframe + Sensors Component
(GENESIS aerodynamics element + 6DOF equations of motion +sensors element)

(iii) Open – Loop Airframe + Sensors + + Atmosphere Component
(GENESIS aerodynamics element + 6DOF equations of motion +sensors element+standard atmosphere element)

(iv) Open – Loop Airfame + propulsion system component
(GENESIS aerodynamics element + 6DOF equations of motion + propulsion element (inlet/nozzle/engine))

Other combinations are possible including virtually any grouping of the elements shown in Fig 2 , however some make no sense, as explained in [4].

GENESIS linearization is based first on the fact that a nonlinear system represented in block diagram form may be linearly represented at a given operating point by doing these things[4]:

(a) Replace each dynamic element in the block diagram with a linear approximation
(b) Replace the interconnecting path network with a new series of linear single input / single output paths. Construct these paths such that each of the possible connections between the external inputs, external outputs, dynamic element inputs and dynamic element outputs is represented by one and only one of the paths. Furthermore, each of the paths should contain its own constant gain factor.

The first step in a GENESIS linearization is then to construct a set of matrix equations that represents the linearized block diagram (this is done automatically by the GENESIS Linear Model Extraction (LME) routine). The resulting set of equations is called the interconnection equations, which may be written in the form:

$$u = F\,y + G\,U \qquad (1)$$
$$Y = H\,y + J\,U$$

where u is a vector containing the inputs to the dynamic elements, y is a vector containing the corresponding dynamic element outputs, U is a vector of external inputs and Y is a vector of external outputs. F,G,H,J are matrices of path gain factors The interconnection equations are not sufficient to completely describe the resulting linear block diagram representation. Equations are also needed to represent the response of the dynamic elements. Individual state space linear system equations are written and then all such state

space systems are appended together to form the dynamic element equations:

$$\dot{x} = A_0 x + B_0 u \qquad (2)$$
$$y = C_0 x + D_0 u$$

Note that once the user has defined linear model inputs and outputs and defined several LME parameters, GENESIS automatically generates the linear model system matrices (A_0, B_0, C_0, D_0) for the component defined to be linearized by use of the LINEARIZE command. A typical example of inputs and outputs for generating a linear model of the open loop airframe might be:

Inputs: Left /Right Tail deflection
Left / Right Aileron deflection
Left / Right Rudder deflection

Outputs: Angle-of Attack, Pitch rate, Altitude, Normal acceleration, Sideslip,Roll rate, Yaw rate

Several linear models were generated on the VAX for the original MAVRIC contract[3]. Linear models of the open loop airfame + engine in a trimmed state were generated from Mach = 10 to 15 (scramjet mode) and from Mach = 16 to 18 (rocket mode) using the newly developed SGI / UNIX GENESIS v2.2 code . Fig 5 shows a comparison of a short dynamic GENESIS nonlinear simulation of the entire vehicle versus a simulation of a linear model of the scramjet + airframe at M=10 and Alt = 120,000ft. Here the system matrices for the linear model are used with Matlab to generate the corresponding step response as generated from GENESIS itself. A .1 degree step is input to the elevons for 5 seconds then removed for purposes of comparison. As the figure indicates, there is excellent agreement between the linear and nonlinear responses.

CONTROL LAW DESIGN

During the original MAVRIC contract numerous control law designs were done . These were broken out based on portions of the trajectory: scramjet ascent, rocket ascent and synergetic plane change. Three nominal design points were chosen for each portion and three other design points were chosen as "off-nominal". H^∞ robust multivariable control synthesis was used to design the controllers.The control law designs were evaluated based on (1) flying qualities (2) stability robustness (3) nominal performance (4) Performance robustness. As noted in [3], which contains a very detailed account of the overall control design process and results, these designs produced good to excellent command following, turbulence rejection and robustness properties.

At the time of this writing, no new control law designs using the linear models generated on the SGI from the v2.2 MAVRIC have been done. All linear models and the full nonlinear simulation are in place to do those future control law designs and evaluate their effect on the vehicle's flying qualities. This new design work is expected to commence later this summer.

MAVRIC SIMULATION EXAMPLES

The task of re-hosting the original VAX/VMS GENESIS v1.96 MAVRIC simulation on SGI/UNIX with updated version 2.2 GENESIS was considered complete (i.e. the new version validated) when sufficient matching of output time-histories between the two simulations was observed. Since a working original version of the simulation currently exists only on the Northrop-Grumman VAX/VMS, simulation results had to be transferred from there for the comparisons with the new simulation at AFRL. Northrop-Grumman manpower constraints limited the number of simulation comparisons available for validation purposes. Figures 6 – 12 show comparisons of AFRL vs Northrop-Grumman time histories and other AFRL simulation results. Since new control law designs have not been done yet for the v2.2 simulation, all results shown here are for open loop vehicle dynamics.

Figure 6 shows a comparison between the original VAX/VMS v1.96 MAVRIC simulation and the new SGI/UNIX v2.2 simulation. After trimming the vehicle at Mach=10 and Altitude = 120,000f, a .1 deg pitch command is issued to the elevons for 5 seconds, then removed. Figure 7 shows a similar comparison for the two simulations , only here linear models of the engine+airframe are used to obtain the time-history response. Note that the linear responses are indistinquishable. The simulations performed to produce figures 8 and 9 are completely analogous to those for figures 6 and 7 respectively, except that figures 8 and 9 show the use of rocket propulsion. Again, the vehicles are trimmed sraight and level (at Mach=16 and Altitude = 150,000ft)

then a .1 deg pitch command is issued to the elevons for 5 seconds, then removed. As for the scramjet simulations at Mach=10, the linear simulations at Mach = 16 show virtually indistinquishable time-histories. Finally, figures 10-12 show the resulting v2.2 simulation time-histories for angle of attack, Mach number and sideslip when the vehicle is trimmed at Mach=13 and altitude= 135,000ft and a 10 degree step in elevon is issued for 5 seconds, then removed. The vehicle's inherent open loop instabilities are clearly present in figures 10 and 12.

CONCLUSIONS

This paper has discussed a new version of the hypervelocity vehicle simulation developed for the 1987-1991 Northrop-Grumman Modern Aerospace Vehicles Robust H-Infinity Control (MAVRIC) contract. The new version is implemented in GENESIS v2.2 on the SGI/UNIX system. The GENESIS software's basic functionality was discussed, along with the simulation's aerodynamic, propulsion and structural models. A discussion of the generation of linear models for the vehicle and issues relating to control law design was followed by several examples of the use of the new simulation and the original in simple pitch-up maneuvers.

With the new version of the MAVRIC simulation now available in AFRL, it is anticipated that work will soon commence in the areas of hypersonic vehicle multidisciplinary control law design and analysis, hypersonic flying qualities analysis, and comparison studies of various advanced modeling techniques for the vehicle's highly coupled aerodynamics, structural dynamics, thermodynamics and propulsion systems. The new GENESIS v2.2 MAVRIC simulation documented in this paper should prove invaluable in this important research.

ACKNOWLEDGMENTS

The author would like to acknowledge the contributions of several individuals in the task of developing the new GENESIS v2.2 MAVRIC simulation on the SGI/UNIX system. Ms Jan DeLuca of Northrop-Grumman provided tremendous assistance in the effort, including suggestions for debugging procedures and numerous input and output files for simulation validation comparisons, model linearization, and vehicle trimming. Mr William Blake and Mr James Simon of AFRL and Ms Meredith Saliers (visiting co-op student from Purdue University) developed an aerodynamic model of the vehicle which is valid at Mach numbers below 10, which the original simulation did not possess. Ms Saliers was the primary contributor to this aspect of the simulation development task, and in addition, made a significant contribution to the formatting of this aerodynamic model in GENESIS and it's integration into the existing v2.2 MAVRIC simulation code.

REFERENCES

[1] Lind, R., Interoffice Memorandum, Air Force Research Laboratory, Flight Dynamics and Control Branch, June 1998.

[2] Haiges, K.R., Tich, E.J., Madden, K.P. "Robust Control Law Development For Modern Aerospace Vehicles Task 1: Model Development",WRDC-TR-89-3080, Northrop Corporation, Aircraft Division, Hawthorne, CA , Aug 1989.

[3] Haiges.K.R., Chiang, R.C., Madden, K.P., Emami-Naeini,A., Safonov,M.G., "Robust Control Law Development For Modern Aerospace Vehicles",WL-TR-91-3105, Northrop Corporation, Aircraft Division, Systems Control Technology Inc, University of Southern California, August 1991

[4] Wong,J.P. et al "Companion Guide to GENESIS Software Version 2.2", Vol I-User's Guide, Vol II - Programmer's Guide, Vol III-Linearization Guide, Northrop Corporation B-2 Division, September 1993.

[5] McKeehen, P.D., "GENESIS Simulation of a Modified VISTA/F-16", AIAA-95-3381-CP, presented at Flight Simulation Technologies Conference, Baltimore, MD, Aug 1995.

[6] McKeehen, P.D., "GENESIS Non-Realtime Simulation of a Tailless Aircraft", AIAA-98-4159-CP, presented at Modeling and Simulation Technologies Conference, Boston, MA, Aug 1998.

[7] Grismer, D.S., Myatt, J.H., McKeehen,P.D., Buffington, J.M., "Application of a Nonlinear-Indicial-Response Model to a Tailless Aircraft

Simulation", AIAA-97-3644-CP, presented at Atmospheric Flight Mechanics Conference, Boston, MA, Aug 1998.

8 Gillard, W.J., Dorsett, K.M., "Directional Control for Tailless Aircraft Using All Moving Wing Tips", Proc. AIAA Atmospheric Flight Mechanics Conference, New Orleans, LA, August 1997.

9 Gentry, A.E., Smyth, D.N., and Oliver, W.R., "The Mark IV Supersonic-Hypersonic Arbitrary Body Program," AFFDL-TR-73-159, November 1973.

10 Williams, J.E., and Vukelich, S.R., "The USAF Stability and Control Digital Datcom," AFFDL-TR-79-3032, April 1979.

11 Nielsen, J., Missile Aerodynamics, McGraw Hill, 1960.

12 Roskam, J., Airplane Design , Roskam Aviation and Engineering Corporation, 1989.

13 Burns, K.A., Deters, K.J., Haley, C.P., and Kihlken, T.A., "Viscous Effects on Complex Configurations," WL-TR-95-3059, August 1995.

FIGURES

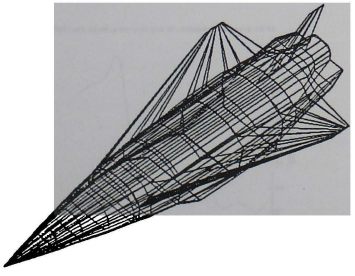


Fig 1 – TSTO Second Stage MAVRIC Configuration

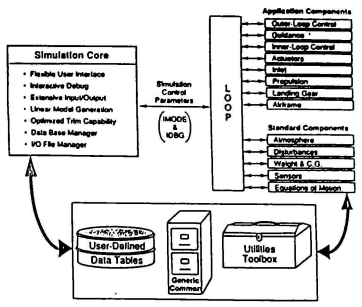


Fig 2 - GENESIS Structure

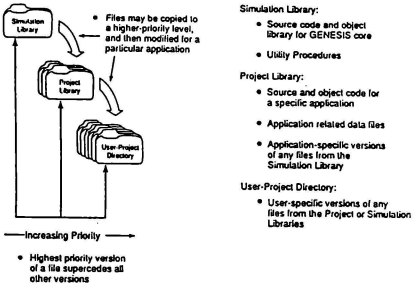


Fig 3 - GENESIS Configuration Management

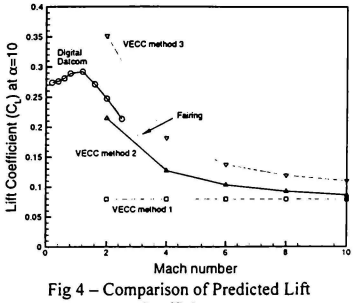


Fig 4 – Comparison of Predicted Lift Coefficients

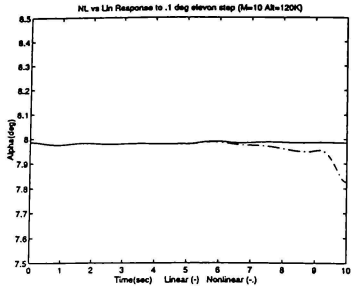


Fig 5 – Scramjet Nonlinear vs Linear Response (M=10 Alt=120,000ft)

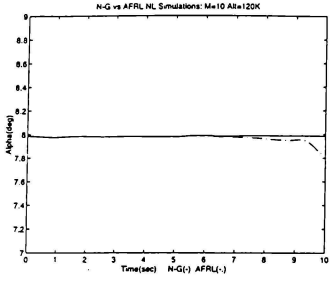


Fig 6 – N-G vs AFRL NL Scramjet Simulation (response to .1 deg elevon deflection at M=10 Alt=120,000ft)

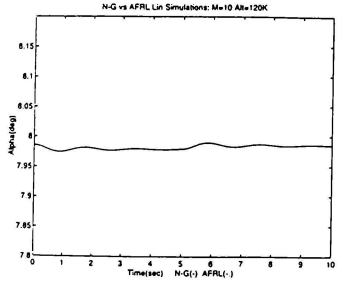


Fig 7 - N-G vs AFRL Lin Scramjet Simulation (response to .1 deg elevon deflection at M=10 Alt=120,000ft)

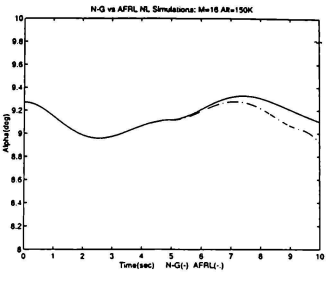


Fig 8 - N-G vs AFRL NL Rocket Simulation (response to .1 deg elevon deflection at M=16 Alt=150,000ft)

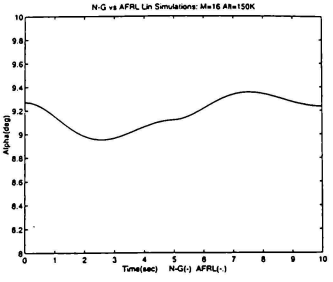


Fig 9 - N-G vs AFRL Lin Rocket Simulation (response to .1 deg elevon deflection at M=16 Alt=150,000ft)

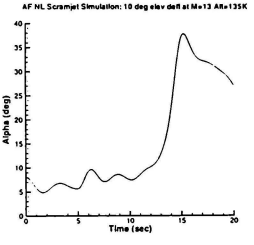


Fig 10 - AOA Time-history: 10 deg elev defl at Mach =13 and Alt=135,000ft

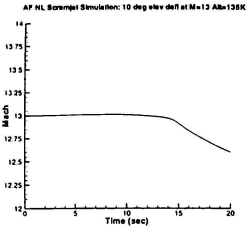


Fig 11 - Mach Time-history: 10 deg elev defl at Mach =13 and Alt=135,000ft

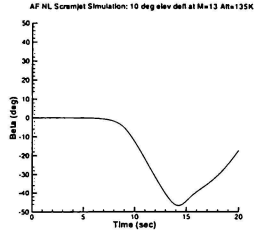


Fig 12 – Sideslip Time-history: 10 deg elev defl at Mach =13 and Alt=135,000ft

AIAA-99-4327

INFLUENCE OF LANDING GEAR DESIGN ON HELICOPTER GROUND RESONANCE

I. Sharf and L. Monterrubio

Department of Mechanical Engineering
University of Victoria
Victoria, B.C., Canada, V8W 3P6
e-mail: isharf@uvic.ca, lmonterr@uvic.ca

ABSTRACT

The objective of this work is to investigate the location of ground resonance instabilities of a helicopter with different skid landing gear designs. Two approaches are proposed to achieve this objective. In the first approach, ANSYS is used to define a detailed finite-element model of a particular landing gear configuration and then, to obtain the mass, stiffness and damping matrices with respect to the center of mass of the aircraft. These matrices are subsequently used in a standard ground resonance mathematical model to calculate the regions of instability of the helicopter. The second approach uses a modal analysis of the fuselage and the rotor systems in ANSYS. The fuselage is modelled in ANSYS as a rigid body sitting on the flexible landing gear. The rotor is represented with a finite element model, which consists of flexible (but stiff) blades interconnected to rigid offsets with lead-lag hinges. Because standard finite elements software such as ANSYS do not have the capability to properly couple the fuselage and the rotor, modal analysis of these two subsystems is carried out separately, thus predicting the uncoupled frequencies only. As illustrated with examples, the first approach gives a better description of the ground resonance instabilities of the helicopter because it predicts their severity, width as well as location. The numerical examples also allow some observations on how different landing gear designs affect ground resonance instabilities of the craft.

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

1. INTRODUCTION

Helicopters with articulated or soft-in-plane rotors may experience the ground resonance problem at certain rotor speeds upon landing. It is very important to avoid this instability in the vicinity of the nominal rotor speed since a severe instability can lead to a catastrophic failure of the craft. Mathematically the ground resonance problem can be attributed to the coupling between a natural frequency of the fuselage with the regressing lead lag mode of the rotor. Several ground resonance models have been developed and used to establish criteria for avoiding this phenomenon. Coleman *et al*[1] developed one of the earliest models. In this model three or more identical blades are mounted on a pylon. The blades are allowed to have deflections in the plane of rotation only while the pylon has two degrees of freedom corresponding to horizontal translations. The pylon is modelled by means of effective masses and stiffnesses. Hence the model is essentially formulated by assuming that the helicopter on its landing gear can be represented by effective parameters applied at the hub. The work by Coleman *et al*[1] has been taken as a reference for most of the subsequent ground resonance models.

Lytwyn *et al*[2] address the subject of ground resonance of low-stiffness hingeless rotors. It is stated that although the source of the instability is purely mechanical, the aerodynamic forces affect it through aeroelastic coupling and damping. The model used in this paper represents the landing gear and the landing surface with two horizontal springs and a vertical spring. Results demonstrate that a vertically soft landing gear can avoid the ground resonance phenomenon over the entire rotor speed range.

Done[3] presents a very simple model, where a rotor with two degrees of freedom is attached to a chassis with only one-degree of freedom. A concentrated mass

attached to a single degree of freedom spring represents the chassis and the rotor consist of blades attached to offsets by a hinge with a rotational spring. Criteria and size of the instability are derived in closed form. Johnson[4] presents the classical ground resonance analysis, which is based on the model with four degrees of freedom. Two degrees of freedom describe the longitudinal and lateral displacements of the hub and two degrees of freedom represent the lead-lag motion of the blades. Johnson[4] also presents results for articulated and soft-in-plane rotors. As in the other papers, results are presented in the form of "Coleman" diagrams in which the natural frequencies and damping of the system are plotted against the angular speed of the rotor.

The ground resonance model employed in our work was taken from the work of Nahas[5]. In this model, the fuselage can have up to six rigid-body degrees of freedom and the landing gear is represented by "resilient" elements. The latter are each comprised of three linear springs to capture the stiffness in three directions and the corresponding dashpots. The rotor system has two degrees of freedom in the plane of rotation representing the location of the center of mass of the rotor and the rotor equations are identical to those in Coleman *et al*[1].

Dick *et al*[6] describe and analyse the ground resonance equations in order to explain the phenomenon in a simpler way and to understand the physics of the problem. In particular, the authors consider a three degree of freedom ground resonance model and analyse in detail the effects of different couplings between the rotor degrees of freedom and fuselage.

The aforementioned references comprise a mere sampling of the literature on the ground resonance phenomenon. A recent comprehensive literature review of this problem can be found in Sharf[7]. Of particular relevance to the subject here is the fact that majority of ground resonance models use a very simplistic discrete representation of the landing gear. Moreover, in contrast to the rotor, the effect of landing gear properties on the ground resonance instabilities has been hardly investigated.

The ultimate goal of the work described in the present manuscript is to develop a model, which will allow one to easily carry out ground resonance analysis of a helicopter with different landing gears. Our motivation for this work, is two-fold. First of all, as already mentioned, the effect of landing gear design on the ground resonance instability has not been investigated comprehensively. Secondly, situations frequently arise in practice where it is necessary to fly the same craft but depending on the particular mission, with different landing gears. To be confident in the safety of such a retrofit, it would be helpful to have ground resonance results for the aircraft with different sets of gear.

To achieve our goal, it is necessary to have a tool, which allows modelling of a helicopter with different landing gears. A finite element representation of a landing gear is typically used for landing gear design as it gives an accurate description of the mass, stiffness and damping characteristics of the gear. The subject of this paper is how such a model can be used for helicopter ground resonance analysis.

Two approaches are proposed in this paper. In the first approach, presented in Section 2, we use one of the commercial finite-elements packages ANSYS to create the substructure of a detailed finite element model of a landing gear. The mass, stiffness and damping matrices of this substructure represent the reduced matrices of the landing gear with respect to the center of mass of the helicopter. These can be directly used in a standard ground resonance mathematical model to find the instability regions of the craft and their severity. As noted earlier, the ground resonance model proposed in Nahas[5] is used in this approach.

The second approach is described in Section 3 of this paper. It involves carrying out modal analysis of the craft directly in ANSYS. Accordingly, models of the fuselage and rotor subsystems are built in ANSYS. The fuselage model involves a substructure representing the landing gear and the inertia properties of the fuselage. The rotor model accounts for changes in its stiffness due to angular velocity by including stress stiffening and spin softening effects. The uncoupled natural frequencies of the fuselage are determined once as they remain constant. The frequencies of the rotor system are determined for different RPM. The locations of ground resonance instabilities are then obtained from the crossings of rotor and fuselage frequencies.

Results of the two approaches for a helicopter with four landing gears are presented in Section 4. Finally, conclusions on the two approaches and the effects for different landing gear configurations are included in Section 5.

2. GROUND RESONANCE ANALYSIS WITH FINITE ELEMENT MODEL OF A LANDING GEAR

As noted in the introduction, the ground resonance model adopted for this approach is taken from the work by Nahas[5]. This model is valid for helicopters with three or more blades. It includes up to six degrees of freedom for the fuselage and two degrees of freedom for the rotor in the plane of rotation. For simplicity the fuselage and blades are modelled as rigid bodies. The rotor system includes an offset and a lead-lag hinge for each blade. The hinges contain rotational springs and dampers for the lead-lag degree of freedom.

The linearized equations of motion of the complete system have the following general form:

$$T\ddot{U} + D\dot{U} + SU = 0 \qquad (1)$$

where T, D and S are the inertia, damping and stiffness matrices and U is the vector of the fuselage and rotor generalised coordinates. The solution of the polynomial eigenvalue problem for the model in equation (1) gives the frequencies of oscillation of the system and their rate of growth or decay. In particular, the imaginary part of the complex conjugate eigenvalues of the system represents the frequencies while the real part gives the damping. Positive damping values indicate a growth in the amplitude of the oscillation of the system, while negative values indicate a decay.

The first six degrees of freedom in U correspond to the degrees of freedom of the fuselage—three translations and three rotations. Accordingly, the 6x6 submatrices of T, D, S represent the properties of the fuselage supported on a landing gear. In Nahas' model, the stiffness and damping matrices of the fuselage are evaluated from the properties of the resilient elements. A resilient element is defined by mutually perpendicular springs and dampers attached to the fuselage at a certain distance from its center of mass.

In the present approach, a detailed finite element model of the landing gear in ANSYS is used to obtain the 6x6 matrix representation of the stiffness and damping properties of the fuselage. This is done by creating a substructure of the finite element model of the landing gear. Since the stiffness and damping properties in model (1) correspond to the center of mass degrees of freedom, the substructure representation of the gear must be defined with respect to the same three translations and three rotations of the center of mass. The resulting substructure gives an accurate 6x6 matrix representation of the stiffness and damping properties of the landing gear. These matrices can be used directly in the ground resonance mathematical model of equation (1).

The main steps of the procedure carried out in ANSYS are listed below:

a) Create a finite element model of the landing gear.
   A skid landing gear consists of two crosstubes and two skids, as shown in the schematic Figure 1. PIPE16 or BEAM4 elements are used in this paper to discretize the landing gear structure. Other type of elements available in ANSYS can also be used. In this model the nodes at the end of the crosstubes are fixed (nodes 2, 11, 18 and 27 in Figure 1). Nodes at the attachment points of the landing gear to the fuselage must be defined in order to define the desired substructure (nodes 39, 42, 59 and 62 in Figure 1).

b) Create a rigid region between the attachment points of the landing gear and the center of mass of the helicopter.
   The CERIG command in ANSYS is used to define rigid regions by implementing appropriate constraint equations. The first node contained in the resulting rigid body is the "master" node and the rest of the nodes are the "slaves" of the rigid body. Note that the degrees of freedom of the "master" node are retained in the analysis while the degrees of freedom of the "slave" nodes are removed.
   The "master" node of the rigid body representing the constraints between the fuselage and the landing gear must be defined at the location of the center of mass of the helicopter in order to obtain the properties of the landing gear with respect to this node (node 73 in Figure 1). The nodes at the attachment points of the landing gear must be the "slave" nodes of the rigid body (nodes 39, 42, 59 and 62 in Figure 1).

c) Create a substructure of the model using the substructure analysis.
   The ANSYS command ANTYPE,SUBSTR specifies the substructure analysis. In this analysis, the six degrees of freedom of the node at the center of mass of the helicopter (node 73 in Figure 1) must be defined as master degrees of freedom. The resulting substructure gives the reduced 6x6 matrix representation of the landing gear with respect to the center of mass of the helicopter.

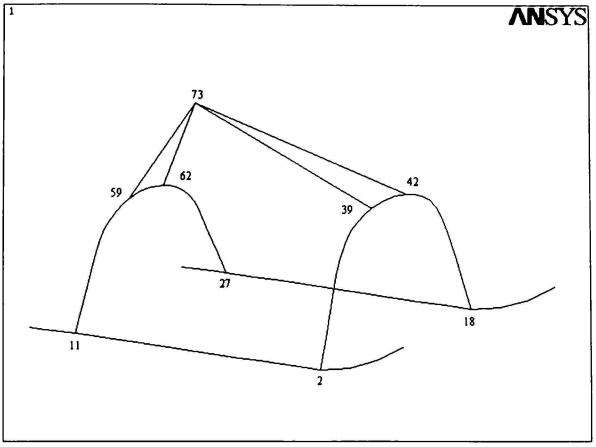


Figure 1. Schematic representation of a skid landing gear rigidly constrained to the center of mass of the helicopter

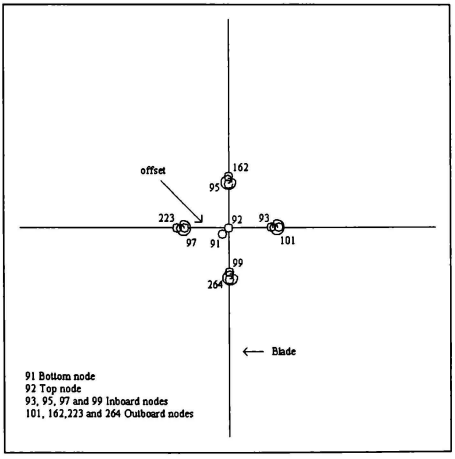


Figure 2. Finite element model of the rotor system.

In this paper, no structural damping of the landing gear is included, but it can be developed by assuming, for example, Rayleigh damping model for the gear. Also, the mass matrix of the landing gear can be extracted from the substructure solution, if desired.

The ground resonance model of equation (1) was implemented in a C code. The inputs to this model are the inertia properties of the fuselage, the geometric and inertial properties of the rotor, and the matrices of the landing gear substructure, as defined by the procedure above. The inertia properties of the helicopter can be described with the total properties of the helicopter (total mass, moments and products of inertia). Changes in this matrix due to a new landing gear design are usually insignificant. In case a higher accuracy in the mass matrix is required, the difference between the mass matrices of the new landing gear design and the original design must be added to the original total mass matrix. Results obtained by using the approach described here for a helicopter with different landing gear designs are presented in Section 4 of this paper.

3. UNCOUPLED MODAL ANALYSIS IN ANSYS

In this approach, modal analysis of the fuselage and the rotor systems is carried out in ANSYS. These two subsystems of the aircraft are treated separately because ANSYS does not have the capability to properly couple them. Accordingly, modal analysis of each subsystem yields the uncoupled frequencies of the rotor and fuselage.

The model of the rotor system is described in the following subsection. The fuselage model consists of the finite element model of the landing gear, as described in Section 2 and the mass properties of the fuselage body. The latter are defined in ANSYS by using a mass element, MASS21, which is placed at the center of mass of the helicopter. The substructure solution and the modal analysis of the fuselage and landing gear are carried out to obtain six natural frequencies of the aircraft.

3.1 Rotor System Model

A four bladed rotor system is modelled in this paper. The model contains a hinge at the center of the rotor, four offsets, four lead-lag hinges and four blades. The offsets are connected to the hinge at the center of the rotor system and to the lead-lag hinges on the outboard side. The blades are then attched to the outboard end of the lead-lag hinge (see Figure 2). The offsets are modelled as massless rigid bodies using the CERIG command. In order to compare the results from this approach with those of the approach described in Section 2, the blades must be rigid as well. However, it was found that for such a model of the rotor, ANSYS is not capable of predicting the variation of the blade lead-lag frequencies with the rotational speed of the rotor. As a result, blades have been modelled with flexible pipe elements, but with a very high stiffness.

Two coincident nodes define the location of the COMBIN7 element in ANSYS which is used to define hinges in the rotor model. These nodes are identified as inboard and outboard when referring to the nodes at the lead-lag hinges, or top and bottom when referring to the hinge at the center of the rotor (see Figure 2). The main steps required to define the rotor system in ANSYS are as follows:

a) Define a hinge by using a COMBIN7 element at the center of the rotor to allow free rotation of the rotor system about the shaft axis (hinge defined with nodes 91 and 92 in Figure 2).
b) Define lead-lag hinges by using COMBIN7 elements, including rotational stiffness and damping at the hinges (hinges defined with node pairs 93/101, 95/162, 97/223 and 99/284 in Figure 2).
c) Define a single rigid body containing the offsets by using the CERIG command. The top node of the center hinge must define the "master" node and the inboard nodes at each lead-lag hinge must define the "slave" nodes of the rigid body.
d) Define blades attached to the outboard nodes of the lead-lag hinges with PIPE16 elements. As noted before, more than one element must define each blade to achieve good accuracy in the solution. Results presented in this paper are for a rotor model with blades discretized with 60 elements.

3.2 Rotor System Modal Solution

Stress stiffening and spin softening effects are included in the substructure analysis, in order to produce the correct changes of the frequencies of the rotor system with angular speed of the rotor. Stress stiffening[8] is used to provide strength in a structure, which normally does not have resistance to loads. This strength is usually caused by a centrifugal force on a rotating body. Spin softening[9] adjusts the stiffness matrix of a rotating body for dynamic mass effects.

The frequency equation[10] used in ANSYS to solve the eigenvalue problem of an undamped system including spin softening and stress stiffening effects is:

$$| (K + K_s - \Omega^2 M) - \omega^2 M | = 0 \qquad (2)$$

where K, K_s and M are the stiffness, stress stiffening and mass matrices; Ω and ω are the angular rate of rotation and the natural circular frequencies of the rotating body.

The following steps are carried out in ANSYS:

a) Define angular rotor velocity about Z-axis with the OMEGA command.
b) Perform static solution including prestress analysis using the PSTRESS,ON command. The prestress analysis must be carried out in order to include the stress stiffening of the blades.
c) Perform substructure analysis including prestress solution and spin softening option KSPIN, in the OMEGA command. The degrees of freedom at each one of the outboard nodes of the lead-lag hinges and the top node at the rotor center are selected as master degrees of freedom. This choice is based on ANSYS recommendation to select master degrees of freedom where low stiffness and high mass or inertia properties are defined. The degrees of freedom at the outboard nodes of the lead-lag hinges are retained to give the frequency of the blade lead-lag motion.
d) Perform modal analysis of the substructure. Several eigensolvers are available in ANSYS. If the substructure includes damping, the damped eigensolver can be invoked by using the MODOPT,DAMP command. Otherwise one of the other methods can be used.

Results obtained by using the approach described here for a helicopter with different landing gear designs are presented in Section 4.2.

4. RESULTS

4.1 Approach 1 Results

We first present the ground resonance results obtained by using the approach described in Section 2. The input data used to get these results are summarized in Table 1, with exception of the stiffness and damping matrices that were obtained with ANSYS. The inertial and geometric parameters specified in Table 1 do not exactly correspond to a particular helicopter. The number of blades, length of blades, offsets, height of the hub and mass correspond to Bell 412 helicopter. The rest of the inertial properties were estimated by scaling values from Kessler *et al*[11] in accordance with the known Bell 412 data. In this paper products of inertia were neglected in order to compare results of the two approaches. MASS21 element used in the second approach does not have the capability to define products of inertia.

Number of blades = 4
Length of the blade = 6.30 [m]
Offset = 0.70 [m]
Mass of blade = 30.00 [kg]
Moment of inertia of blade, I_b= 396.90 [Kg m^2]
Lead-lag stiffness coefficient, K_b = 4,955.00 [N m]
Lead-lag damper coefficient = 60 [N m s]
Mass of aircraft = 4,500.00 [kg]
$IXX_{aircraft}$ = 3,600.00 [Kg m^2]
$IYY_{aircraft}$ = 11,600.00 [Kg m^2]
$IZZ_{aircraft}$ = 9,200.00 [Kg m^2]
Distance of hub above C. G. of aircraft = 1.70 [m]

Table 1. Input data for example aircraft.

The blade lead-lag stiffness K_b in Table 1 was estimated from the equation below derived from relations presented by Johnson[4].

$$K_b = I_b \left(\frac{rpm_{no\,min\,al}}{2\pi} \right)^2 \left((1-e) v_\zeta^{\,2} - 1.5e \right) \qquad (3)$$

where:

e = offset length divided by blade radius,

v_ζ = rotating natural frequency of blade fundamental lag mode. The value for v_ζ was set to 0.8 [1/rev] which corresponds to a typical value for hingeless rotors. The blade lead-lag damping coefficient in Table 1was taken directly from Kessler *et al*[11]. We also note that the nominal speed for the Bell 412 helicopter, estimated from tip velocity of the blades is 34.00 [rad/s].

Four landing gear designs were considered in our analysis. These are defined by the heights of the crosstubes as given in Table 2.

Landing gear	height of crosstubes	
	front [m]	aft [m]
Standard	0.57	0.49
High	0.72	0.66
Extra high	1.02	0.99
Special Gear	0.89	0.92

Table 2. Heights of landing gears used in this paper.

The ground resonance analysis was carried out for the helicopter defined above and each of the four landing gear designs. Results are presented in Figures 3-6 in the form of "Coleman" diagrams. Instability regions are clearly identified from these figures. The maximum severity of instabilities and the corresponding rotor speeds are summarised in Table 3.

Landing Gear	Max. Severity [rad/s]	Rotor Speed [rad/s]
Standard		
#1*	1.86	38.11
#2	0.96	80.00
High		
#1*	1.67	34.75
#2	1.07	70.79
Extra High		
#1	0.52	22.26
#2	1.33	30.67
#3	0.93	59.58
Special Gear		
#1	0.61	24.58
#2	1.43	32.27
#3	0.93	61.34

Table 3. Approach 1 results.

*Two degrees of freedom of the fuselage couple with the regressing lead-lag mode of the rotor in this region of instability.

4.2 Approach 2 Results

The same fuselage and rotor systems as used in the previous section were used to obtain the results presented in Figures 7-10. As described in Section 3, the second approach yields uncoupled frequencies of the fuselage and rotor systems. In uncoupled analysis, the fuselage frequencies are independent of the rotor speed and hence are shown as straight horizontal lines in the frequency plots. The lead-lag frequencies obtained in ANSYS are not the cyclic frequencies of the rotor. To locate the regions of ground resonance instabilities, these frequencies must be transformed to the frequencies of the cyclic rotor modes. Denoting by Γ the physical lead-lag frequency calculated by ANSYS, the cyclic frequencies for an uncoupled rotor can be determined from the following equations:

$$\omega_{regressing} = |\Gamma - \Omega| \quad (4)$$

$$\omega_{progressing} = \Gamma + \Omega \quad (5)$$

The location of the instabilities from the plot of the uncoupled frequencies is defined by the rotor angular speed at which the regressing mode frequency intersects one of the natural frequencies of the fuselage. These results are summarised in Table 4.

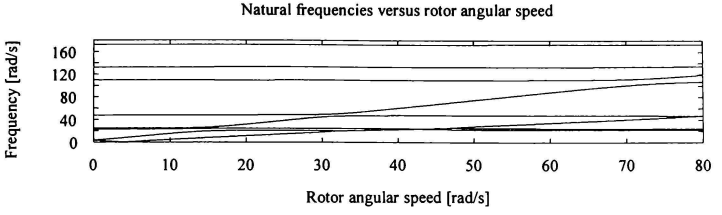
Landing Gear	Instability Rotor Speed [rad/s]
Standard	
#1	39.07
#2	41.03
#3	78.52
High	
#1	32.38
#2	34.50
#3	69.10
Extra High	
#1	22.48
#2	30.20
#3	58.52
Special Gear	
#1	24.91
#2	31.75
#3	59.94

Table 4. Approach 2 results.

4.3 Discussion of Results

We will first comment on the results obtained with the two procedures. Approach 1 predicts two regions of instability for standard and high landing gears. The second of these occurs significantly above nominal RPM and normally would not be cause for concern. The first region represents an overlap of two instability regions which are identified separately from the uncoupled frequencies of approach 2. The overlap in turn results because two of the aircraft frequencies are in close vicinity of each other. Other than this difference, the uncoupled analysis in ANSYS gives a fairly accurate estimate for the location of instabilities (see results for extra high and special gear). Of course, as was alluded to earlier, the second approach does not give any indication on the severity and width of the instabilities.

The following observations can be drawn by comparing ground resonance results for different landing gears. Lower landing gear designs tend to have higher stiffness which in turn results in higher natural frequencies of the craft. The coupling of the stiffer fuselage with the regressing lead-lag mode will therefore happen at a higher rotor speed. It follows that if the critical instability occurs below the nominal RPM of the craft with a particular landing gear, it is likely that "softening" the gear will alleviate the ground



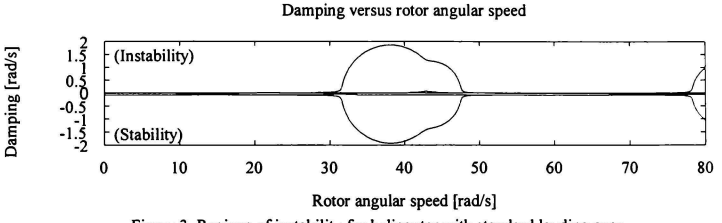
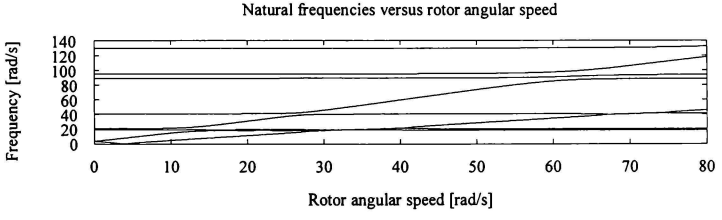


Figure 3. Regions of instability for helicopter with standard landing gear.



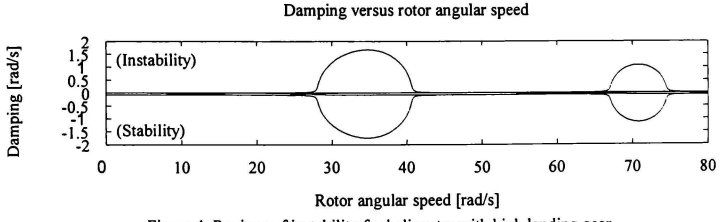
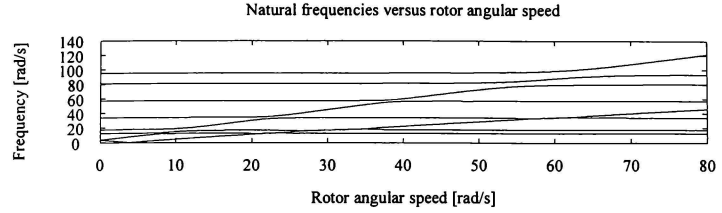


Figure 4. Regions of instability for helicopter with high landing gear.



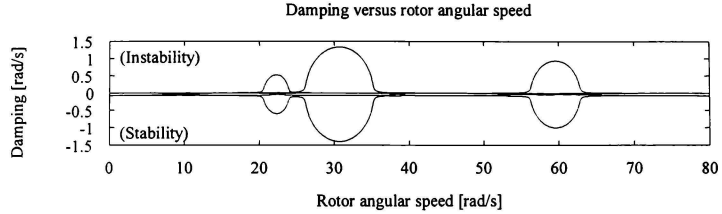
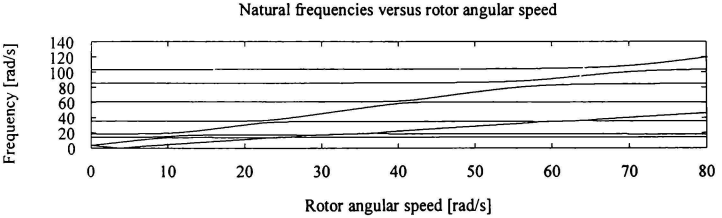


Figure 5. Regions of instability for helicopter with extra high landing gear.



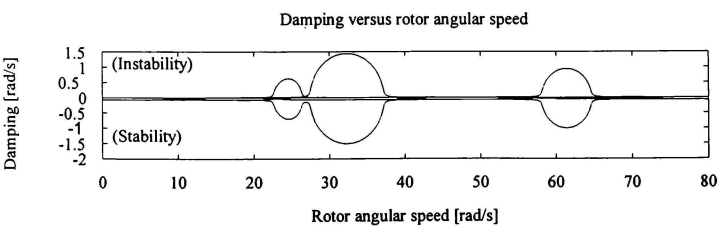


Figure 6. Regions of instability for helicopter with special landing gear.

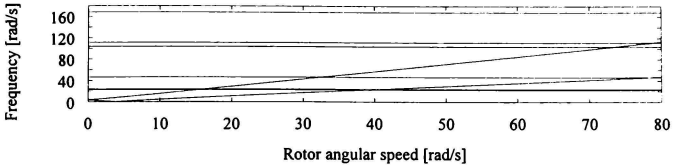


Figure 7. Uncoupled frequencies of fuselage and rotor for helicopter with standard landing gear.

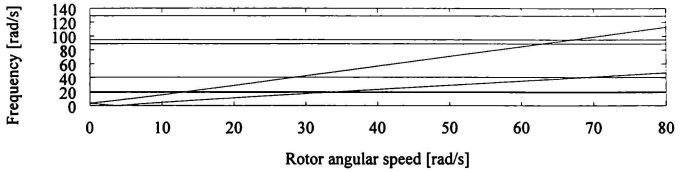


Figure 8. Uncoupled frequencies of fuselage for helicopter with high landing gear.

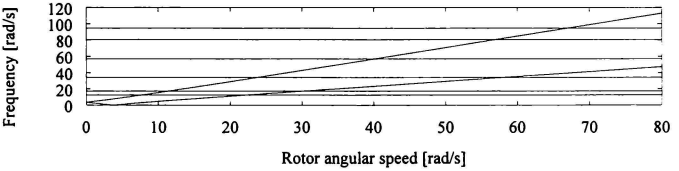


Figure 9. Uncoupled frequencies of fuselage and rotor for helicopter with extra high landing gear.

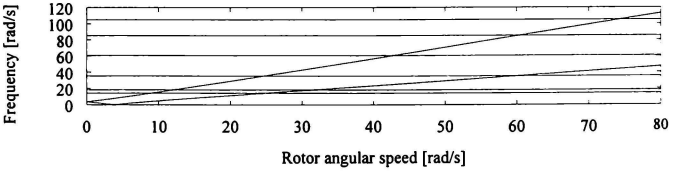


Figure 10. Uncoupled frequencies of the fuselage and rotor for helicopter with special landing gear.

resonance problem. This observation is in agreement with the conclusion reached by Lytwyn *et al*[2] and applies to most situations. However, the opposite holds true if the instability occurs above (but close) to the nominal RPM. In addition to changing the actual frequencies of the aircraft, different landing gear designs may affect the separation of the natural frequencies. In case when the ground resonance instability results from the coalescence with two closely-spaced body frequencies, it tends to be more severe and wide. It is difficult, however, to suggest general guidelines for avoiding such situations. We suggest to investigate them individually, with coupled ground resonance analysis of approach 1.

5. CONCLUSIONS

In this paper, two methodologies for ground resonance analysis of a helicopter have been developed. Both are motivated by the need to have an accurate representation of the helicopter stiffness and damping characteristics. In the first approach, these are obtained from a detailed finite-element model of the landing gear and are then used in a standard ground resonance model. The second approach performs a modal analysis of the uncoupled subsystems representing the fuselage and the rotor. In the uncoupled analysis, the crossing points of the regressing lead-lag frequencies with the fuselage frequencies define the regions of the ground resonance instabilities.

The location of the instabilities found in the second approach is fairly close to that obtained in the first approach. However, the severity and width of the instabilities cannot be obtained with this methodology because ANSYS does not have an automatic capability to properly couple the fuselage and rotor subsystems.

Results for different height landing gears also indicate that higher landing gear designs may alleviate the ground resonance instability, when it occurs below the nominal rotor speed. However, simple design guidelines are not full-proof and in general, coupled ground resonance is warranted.

ACKNOWLEDGMENTS

Funding support from NSERC through Research Grant program and from Dart Aerospace Ltd. is acknowledged. Technical support from Dart Aerospace Ltd. is also acknowledged.

REFERENCES

1.Coleman R. P., Feingold A. M., "Theory of self-excited mechanical oscillations of helicopter rotors with hinged blades", NACA-report 1351, 1958.

2. Lytwyn R.T., Miao W. and Woitsch W., "Airborne and ground resonance of hingeless rotors", Journal of the American Helicopter Society, Vol. 16, No. 2, 1971, pp. 2-9.

3. Done G. T. S., "A simplified approach to helicopter ground resonance", Aeronautical Journal, Vol. 78, May 1974, pp. 204-208.

4. Johnson W, Helicopter Theory. Princeton University Press, Princeton, 1980.

5. Nahas M. N., "Helicopter ground resonance – a spatial model analysis", Aeronautical Journal, Vol. 88, August/September 1984, pp. 299-308.

6. Dick E., Sioen H., Zeoli D., "A basic analysis of helicopter ground resonance", European Journal of Mechanical Engineering, June 1994, pp. 91-101.

7. Sharf I., "Literature survey on helicopter ground resonance", report submitted to Dart Aerospace Ltd, November 1995.

8. ANSYS Release 5.5 User's Manual, "Advanced analysis technique", 1998.

9. ANSYS Release 5.5 User's Manual, "Structural analysis guide", 1998.

10. ANSYS Release 5.5 User's Manual, "Theory reference", 1998.

11. Kessler Ch., Reichert G., "Active control of ground and air resonance including transition from ground to air", European rotorcraft forum 1994, Vol. 20, No. 3, pp. 64.

Developments in Human Centered Cueing Algorithms for Control of Flight Simulator Motion Systems

Robert J. Telban* and Frank M. Cardullo†
State University of New York
Binghamton, New York

Jacob A. Houck‡
NASA Langley Research Center
Hampton, Virginia

Abstract

The authors conducted further research with cueing algorithms for control of flight simulator motion systems. A variation of the so-called optimal algorithm was formulated using simulated aircraft angular velocity input as a basis. Models of the human vestibular sensation system, i.e. the semicircular canals and otoliths, are incorporated within the algorithm. Comparisons of angular velocity cueing responses showed a significant improvement over a formulation using angular acceleration input. Results also compared favorably with the coordinated adaptive washout algorithm, yielding similar results for angular velocity cues while eliminating false cues and reducing the tilt rate for longitudinal cues. These results were confirmed in piloted tests on the current motion system at NASA-Langley, the Visual Motion Simulator (VMS). Proposed future developments by the authors in cueing algorithms are revealed. The new motion system, the Cockpit Motion Facility (CMF), where the final evaluation of the cueing algorithms will be conducted, is also described.

* PhD. Candidate, Department of Mechanical Engineering, Student Member AIAA
† Associate Professor, Department of Mechanical Engineering, Associate Fellow AIAA
‡ Systems Development Branch, Associate Fellow AIAA

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

Introduction

While a visual system alone can provide motion cues at low frequency, physical motion stimuli are necessary to provide higher frequency cues in the range sensitive to the vestibular and somatosensory systems. The addition of high fidelity motion cues from a moving platform in conjunction with visual motion cues have been shown to produce a rapid onset of vection, or the illusion of motion, thus reducing the delay associated with visual motion alone.

A key element in providing physical stimuli in flight simulators is the cueing algorithm that produces the drive signals used to control the motion system hardware. Two viable approaches to motion cueing algorithm development have been identified from research conducted by Wu and Cardullo.[1, 2]

The first technique is a modification of the coordinated adaptive washout algorithm developed by Parrish, et al., hereafter referred to as the "adaptive algorithm".[3] This algorithm uses both first and second order linear washout filters in conjunction with an optimization method that adjusts the filter gains in real time by minimizing the error between the simulated vehicle and the motion platform responses. This methodology effectively produces a set of nonlinear washout filters.

The second technique is the "optimal algorithm" based on that which was developed by Sivan, et al.[4] and later implemented by Reid and Nahon.[5] This algorithm uses higher order filters that are developed, prior to real time application, using optimal control methods. This method incorporates a mathematical model of the human vestibular system, constraining the sensation error between the simulated aircraft and motion platform dynamics.

In their research Wu and Cardullo[1,2] made several modifications to the optimal algorithm implemented by Reid and Nahon[4], resulting in improved performance.

The center of rotation of the motion platform was moved from the pilot's head to the motion base centroid, reducing actuator extension lengths during simulation. In the algorithm development additional states were added to the cost function to enable more flexibility in tuning the algorithm. A nonlinear gain algorithm was developed that scales the aircraft inputs by a third-order polynomial, maximizing the motion cues while remaining within the operational limits of the motion system.

The question has arisen as to what aircraft and simulator control inputs are the most appropriate for the optimal algorithm. The previous developments[4,5] centered on a control input for either the longitudinal or lateral mode with linear acceleration and angular displacement as control inputs. Wu[2] developed an approach using linear acceleration and angular acceleration for the longitudinal mode. This approach shows advantages in controlling additional motion states that were not available in the original development. In addition, since the semicircular canals behave as a transducer for angular velocity input in the range of normal head movements,[6] an approach using angular velocity as input may also be desired.

In this paper an optimal algorithm based on simulated aircraft angular velocity inputs is discussed. Models of the human vestibular system, i.e. the semicircular canals and otoliths, are incorporated within the algorithm in order to constrain vestibular sensation errors. A set of cueing filters is optimized and generated prior to real time implementation.

Motion platform responses generated by this revised optimal algorithm are compared with responses from the optimal algorithm based on angular acceleration input. An objective comparison of motion platform responses and pilot sensation responses (as computed from the vestibular models employing platform motion as a stimulus) was made with responses generated from the adaptive algorithm. The algorithms were then tested on the NASA Langley Visual Motion Simulator in a series of piloted test maneuvers.

Visual Motion Simulator

The Visual Motion Simulator (VMS), shown in Figure 1, is a general-purpose simulator consisting of a two-crewmember cockpit mounted on a 60-inch stroke six-degree-of-freedom synergistic motion base[7,8]. Motion cues are provided in the simulator by the relative extension or retraction of the six hydraulic actuators of the motion base. Both the adaptive and optimal algorithms were used to drive the motion base during this study.

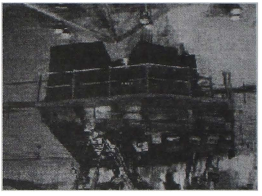


Figure 1. Visual Motion Simulator (VMS).

The cockpit of the VMS, shown in Figure 2, is designed to accommodate a generic transport aircraft configuration on the left side and a generic fighter or rotorcraft configuration on the right side. Both sides of the cockpit are outfitted with three heads-down CRT displays (primary flight display, navigation/map display, and engine display), a number of small standard electromechanical circular instruments and a control display unit mounted in the center. The left side contains a two-axis side stick control loader, and the right side contains a two-axis center stick. Both sides contain control loaded rudder systems. A center aisle stand with throttle quadrant is also available. The cockpit is outfitted with four collimated window display systems to provide an out-the-window visual scene which is driven by an Evans and Sutherland ESIG 3000/GT computer generated image system. The left side of the cockpit was used during the cueing algorithm evaluation study.



Figure 2. Visual Motion Simulator Cockpit.

The simulator includes a nonlinear mathematical model of a Boeing 737-100 aircraft, complete with landing gear dynamics, gust and wind models, radio navigation system models, and instrument and microwave landing system models.[9]

Algorithm Development

In developing an optimal washout filter, the problem is to determine a matrix of linear transfer functions **W(s)** that relates the simulator motion input to the aircraft motion input so that a cost function constraining both the sensation error between the aircraft and simulator pilot is minimized. The structure of the problem is illustrated in Figure 3.

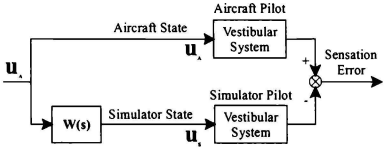


Figure 3. Aircraft Simulation Problem Structure.

A mathematical model of the human vestibular system is used in the filter development. The optimal algorithm described below generates the optimized transfer functions **W(s)** by an off-line MATLAB™ and SIMULINK™ setup. **W(s)** is then implemented on-line. **W(s)** will relate the simulator motion input to the aircraft motion input by $\mathbf{u}_s = \mathbf{W(s)} \times \mathbf{u}_a$. The simulator states $\mathbf{u}_s$ are then used to generate the desired motion base commands.

The filters for four modes: longitudinal (pitch/surge), lateral (roll/sway), yaw, and heave are designed separately in the optimal algorithm development. The algorithm development with angular velocity input for the longitudinal mode is given below. The control input **u** is formulated as

$$\mathbf{u} = \begin{bmatrix} \dot{\theta} \\ a_x \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1}$$

The sensed rotational motion (pitch) is then related to the input u_1 by the semicircular canals model[5,10]

$$\hat{q} = \frac{G_s \tau_a s^2 (1 + \tau_L s)}{(1 + \tau_a s)(1 + \tau_1 s)(1 + \tau_2 s)} u_1 \tag{2}$$

Note that the short time constant τ_2, equal to 0.005 seconds,[11] must be included in the model, otherwise the system equation becomes non-realizable. τ_2 was neglected by Wu^2 in the optimal algorithm formulation based on angular acceleration input. Eqn. (2) can be rewritten as

$$\hat{q} = \frac{T_3 s^3 + T_4 s^2}{s^3 + T_2 s^2 + T_1 s + T_0} u_1 \tag{3}$$

where

$$T_0 = \frac{1}{\tau_a \tau_1 \tau_2}, \quad T_1 = \frac{\tau_a + \tau_1 + \tau_2}{\tau_a \tau_1 \tau_2},$$
$$T_2 = \frac{\tau_1 \tau_2 + \tau_a (\tau_1 + \tau_2)}{\tau_a \tau_1 \tau_2},$$
$$T_3 = G_s \tau_a \tau_L T_0, \quad T_4 = G_s \tau_a T_0$$

™ MATLAB and SIMULINK are registered trademarks of the Mathworks, Inc., 24 Prime Park Way, Natick, MA 01760.

and can be defined in state space notation as

$$\begin{aligned}\mathbf{x}_{1\sim3} &= [x_1 \quad x_2 \quad x_3]^{\mathrm{T}} \\ \dot{\mathbf{x}}_{1\sim3} &= \mathbf{A}_{\mathrm{scc}}\mathbf{x}_{1\sim3} + \mathbf{B}_{\mathrm{scc}}\mathbf{u} \\ \hat{q} &= \mathbf{C}_{\mathrm{scc}}\mathbf{x}_{1\sim3} + \mathbf{D}_{\mathrm{scc}}\mathbf{u}\end{aligned} \tag{4}$$

where in canonical observer form

$$\mathbf{A}_{\mathrm{scc}} = \begin{bmatrix} -T_2 & 1 & 0 \\ -T_1 & 0 & 1 \\ -T_0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B}_{\mathrm{scc}} = \begin{bmatrix} T_4 - T_2T_3 & 0 \\ -T_1T_3 & 0 \\ -T_0T_3 & 0 \end{bmatrix}$$

$$\mathbf{C}_{\mathrm{scc}} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{D}_{\mathrm{scc}} = \begin{bmatrix} T_3 & 0 \end{bmatrix}$$

The sensed specific force (in the longitudinal axis) is now related to the aircraft specific force f_x by the otolith model[5,11,12]

$$\hat{f}_x = G_{o2}\frac{(s + A_o)}{(s + B_o)(s + B_1)} f_x \tag{5}$$

For the center of rotation at the centroid of the motion platform, the specific force is

$$f_x = a_x + g\theta + R_{Sz}\ddot{\theta} \tag{6}$$

where R_{Sz} is the radius from the motion platform centroid to the pilot's head. In terms of the control inputs u_1 and u_2, Eqn. (6) can be transformed into the Laplace domain

$$f_x(s) = u_2(s) + \left(g\frac{1}{s} - R_{Sz}s\right)u_1(s) \tag{7}$$

Substituting Eqn. (7) into Eqn. (5) and rearranging results in

$$\hat{f}_x = G_{o2} \times \left[\frac{-R_{Sz}s^3 - R_{Sz}A_0s^2 + gs + gA_0}{s(s + B_0)(s + B_1)} \quad \frac{s + A_0}{(s + B_0)(s + B_1)}\right]\mathbf{u} \tag{8}$$

Note that in Eqn. (8) the system equation becomes realizable with the inclusion of the otolith break frequency B_1, which was neglected by both Reid and Nahon[4] and Wu[2] in their respective optimal algorithm formulations. Rearranging Eqn. (8) and taking derivatives on both sides results in the differential equation

$$\begin{aligned}&\dddot{\hat{f}}_x + (B_0 + B_1)\ddot{\hat{f}}_x + B_0B_1\dot{\hat{f}}_x = \\ &G_{o2} \times \{R_{Sz}(B_0 + B_1 - A_0)\dot{u}_1 + (g + R_{Sz}B_0B_1)u_1 \\ &\quad + gA_0\int u_1 dt + \dot{u}_2 + A_0u_2\}\end{aligned} \tag{9}$$

which can be rewritten as

$$\begin{aligned}&\dddot{\hat{f}}_x + a\ddot{\hat{f}}_x + b\dot{\hat{f}}_x = \\ &c\dot{u}_1 + du_1 + e\int u_1 dt + f\dot{u}_2 + gu_2\end{aligned} \tag{10}$$

and can then be defined in state space notation as

$$\begin{aligned}\mathbf{x}_{4\sim8} &= [x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8]^{\mathrm{T}} \\ \dot{\mathbf{x}}_{4\sim8} &= \mathbf{A}_{\mathrm{oto}}\mathbf{x}_{4\sim8} + \mathbf{B}_{\mathrm{oto}}\mathbf{u} \\ \hat{f} &= \mathbf{C}_{\mathrm{oto}}\mathbf{x}_{4\sim8} + \mathbf{D}_{\mathrm{oto}}\mathbf{u}\end{aligned} \tag{11}$$

where

$$\mathbf{A}_{\mathrm{oto}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -b & -a & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -b & -a \end{bmatrix},$$

$$\mathbf{B}_{\mathrm{oto}} = \begin{bmatrix} c & 0 \\ d - ac & 0 \\ e & 0 \\ 0 & f \\ 0 & h - af \end{bmatrix},$$

$$\mathbf{C}_{\mathrm{oto}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{D}_{\mathrm{oto}} = G_{O2}\begin{bmatrix} -R_{Sz} & 0 \end{bmatrix}$$

The derivatives of the control input **u** are absorbed into the state space representation in Eqn. (11) by a method given in Brogan.[13] The state space representations in Eqns. (4) and (11) are then combined to form a single representation for the human motion sensation model:

$$\begin{aligned}\dot{\mathbf{x}}_{1\sim8} &= \mathbf{A}_v\mathbf{x}_{1\sim8} + \mathbf{B}_v\mathbf{u} \\ \hat{\mathbf{y}}_1 &= \mathbf{C}_v\mathbf{x}_{1\sim8} + \mathbf{D}_v\mathbf{u}\end{aligned} \tag{12}$$

where

$$\mathbf{A}_v = \begin{bmatrix}\mathbf{A}_{scc} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{oto}\end{bmatrix},\ \mathbf{B}_v = \begin{bmatrix}\mathbf{B}_{scc} \\ \mathbf{B}_{oto}\end{bmatrix},$$
$$\mathbf{C}_v = \begin{bmatrix}\mathbf{C}_{scc} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{oto}\end{bmatrix},\ \mathbf{D}_v = \begin{bmatrix}\mathbf{D}_{scc} \\ \mathbf{D}_{oto}\end{bmatrix}$$

It is assumed that the same sensation model can be applied to both the pilot in the aircraft and the pilot in the simulator as shown in Figure 1. We then define the state error $\mathbf{x}_e = \mathbf{x}_s - \mathbf{x}_a$ and the pilot sensation error **e**, resulting in

$$\begin{aligned}\dot{\mathbf{x}}_e &= \mathbf{A}_v\mathbf{x}_e + \mathbf{B}_v\mathbf{u}_s - \mathbf{B}_v\mathbf{u}_a \\ \mathbf{e} &= \mathbf{C}_v\mathbf{x}_e + \mathbf{D}_v\mathbf{u}_s - \mathbf{D}_v\mathbf{u}_a\end{aligned} \tag{13}$$

It is also necessary for the control algorithm to explicitly access motion states such as the linear velocity and displacement of the simulator, which are desired to appear in the cost function. For this purpose additional terms are included in the state equations

$$\begin{aligned}\mathbf{x}_d &= \begin{bmatrix}x_9 & x_{10} & x_{11} & x_{12}\end{bmatrix}^T \\ &= \begin{bmatrix}\iiint a_x dt^3 & \iint a_x dt^2 & \int a_x dt & \theta\end{bmatrix}^T \\ \dot{\mathbf{x}}_d &= \mathbf{A}_d\mathbf{x}_d + \mathbf{B}_d\mathbf{u}\end{aligned} \tag{14}$$

where

$$\mathbf{A}_d = \begin{bmatrix}0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0\end{bmatrix},\ \mathbf{B}_d = \begin{bmatrix}0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0\end{bmatrix}$$

Input u consists of filtered white noise, and can be expressed in state space as

$$\begin{aligned}\dot{\mathbf{x}}_n &= \mathbf{A}_n\mathbf{x}_n + \mathbf{B}_n\mathbf{w} \\ \mathbf{u} &= \mathbf{x}_n\end{aligned} \tag{15}$$

where

$$\mathbf{A}_n = \begin{bmatrix}-1.0 & 0 \\ 0 & -4\pi\end{bmatrix},\ \mathbf{B}_n = \begin{bmatrix}1.0 \\ 1.0\end{bmatrix}$$

The state equations given in Eqns. (13), (14), and (15) can be combined to form the desired system equation

$$\begin{aligned}\mathbf{x} &= \begin{bmatrix}\mathbf{x}_e & \mathbf{x}_d & \mathbf{x}_n\end{bmatrix}^T \\ \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}_s + \mathbf{H}\mathbf{w} \\ \mathbf{y} &= \begin{bmatrix}\mathbf{e} & \mathbf{x}_d\end{bmatrix}^T = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}_s\end{aligned} \tag{16}$$

where

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_v & \mathbf{0} & -\mathbf{B}_v \\ \mathbf{0} & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_n\end{bmatrix},\ \mathbf{B} = \begin{bmatrix}\mathbf{B}_v \\ \mathbf{B}_d \\ \mathbf{0}\end{bmatrix}$$
$$\mathbf{C} = \begin{bmatrix}\mathbf{C}_v & \mathbf{0} & -\mathbf{D}_v \\ \mathbf{0} & \mathbf{I} & \mathbf{0}\end{bmatrix},\ \mathbf{H} = \begin{bmatrix}\mathbf{0} \\ \mathbf{0} \\ \mathbf{B}_n\end{bmatrix}$$
$$\mathbf{D} = \begin{bmatrix}\mathbf{D}_v & \mathbf{0}\end{bmatrix}$$

with the cost function

$$J = E\left\{\int_{t_0}^{t_1} (\mathbf{e}^T\mathbf{Q}\mathbf{e} + \mathbf{x}_d^T\mathbf{R}_d\mathbf{x}_d + \mathbf{u}_s^T\mathbf{R}\mathbf{u}_s)dt\right\} \tag{17}$$

where E is the average or expected value. Eqn. (17) implies that three variables are to be constrained in the cost function: the sensation error $\mathbf{e}$ along with the additional terms $\mathbf{x_d}$ and $\mathbf{u_s}$ which together define the linear and angular motion of the platform.

The system equation of Eqn. (16) and the cost function of Eqn. (17) can be transformed to the standard optimal control form[14] by the following equations:

$$\dot{\mathbf{x}} = \mathbf{A}'\mathbf{x} + \mathbf{B}\mathbf{u}' + \mathbf{H}\mathbf{w}$$
$$J = E\left\{\int_{t_0}^{t_1} (\mathbf{x}^T\mathbf{R}_1'\mathbf{x} + \mathbf{u}'^T\mathbf{R}_2\mathbf{u}')dt\right\} \tag{18}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_d \end{bmatrix},$$
$$\mathbf{R}_1 = \mathbf{C}^T\mathbf{G}\mathbf{C},\ \mathbf{R}_2 = \mathbf{R} + \mathbf{D}^T\mathbf{G}\mathbf{D},$$
$$\mathbf{A}' = \mathbf{A} - \mathbf{B}\mathbf{R}_2^{-1}\mathbf{R}_{12}^T,\ \mathbf{u}' = \mathbf{u}_s + \mathbf{R}_2^{-1}\mathbf{R}_{12}^T\mathbf{x},$$
$$\mathbf{R}_1' = \mathbf{R}_1 - \mathbf{R}_{12}\mathbf{R}_2^{-1}\mathbf{R}_{12}^T$$

The cost function of Eqn. (18) is minimized when

$$\mathbf{u}' = -\mathbf{R}_2^{-1}\mathbf{B}^T\mathbf{P}\mathbf{x} \tag{19}$$

and defining the optimal feedback gain matrix $\mathbf{F}$,

$$\mathbf{F} = \mathbf{R}_2^{-1}(\mathbf{B}^T\mathbf{P} + \mathbf{R}_{12}^T) \tag{20}$$

$\mathbf{F}$ can then be partitioned corresponding to the partition of $\mathbf{x}$ in Eqn. (16):

$$\mathbf{u}_s = -\begin{bmatrix} \mathbf{F}_1 & \mathbf{F}_2 & \mathbf{F}_3 \end{bmatrix}\begin{bmatrix} \mathbf{x}_e \\ \mathbf{x}_d \\ \mathbf{x}_n \end{bmatrix} + \begin{bmatrix} \mathbf{B}_v \\ \mathbf{B}_d \end{bmatrix}\mathbf{u}_s \tag{23}$$

Remove the states corresponding to the $\mathbf{x_n}$ partition from Eqn. (16):

$$\begin{bmatrix} \dot{\mathbf{x}}_e \\ \dot{\mathbf{x}}_d \end{bmatrix} = \begin{bmatrix} \mathbf{A}_v & \mathbf{0} & -\mathbf{B}_v \\ \mathbf{0} & \mathbf{A}_d & \mathbf{0} \end{bmatrix}\begin{bmatrix} \mathbf{x}_e \\ \mathbf{x}_d \\ \mathbf{x}_n \end{bmatrix} + \begin{bmatrix} \mathbf{B}_v \\ \mathbf{B}_d \end{bmatrix}\mathbf{u}_s \tag{24}$$

After taking the Laplace transform of Eqns. (23) and (24), the following equations are obtained:

$$\mathbf{u_a(s)} = \mathbf{W(s)}\ \mathbf{u_s(s)} \tag{25}$$

where

$$\mathbf{W(s)} = \begin{bmatrix} \mathbf{F}_1 & \mathbf{F}_2 \end{bmatrix} \times \begin{bmatrix} \mathbf{sI} - \mathbf{A}_v + \mathbf{B}_v\mathbf{F}_1 & \mathbf{B}_v\mathbf{F}_2 \\ \mathbf{B}_d\mathbf{F}_1 & \mathbf{sI} - \mathbf{A}_d + \mathbf{B}_d\mathbf{F}_2 \end{bmatrix}^{-1} \times \begin{bmatrix} \mathbf{B}_v(\mathbf{I} + \mathbf{F}_3) \\ \mathbf{B}_d\mathbf{F}_3 \end{bmatrix} - \mathbf{F}_3$$

The optimal filter matrix **W(s)** is computed using a set of MATLAB scripts. The weighting matrices **Q**, **R**, and $\mathbf{R_d}$ given in the cost function of Eqn. (17) are selected and adjusted to produce the desired platform responses. From these weights and the vestibular models the standard optimal control matrices of Eqn. (18) are computed. The algebraic Riccati equation of Eqn. (19) is solved with the MATLAB function **"care"**. The solution for **W(s)** in Eqn. (25) is then computed. Common poles and zeroes are cancelled in each transfer function, yielding a set of seventh-order filters for the longitudinal mode. These filters are then used in a

SIMULINK model that generates the linear acceleration and angular velocity responses.

Algorithm Evaluation

Comparisons of longitudinal responses are made between the optimal algorithm with angular velocity input and the optimal algorithm with angular acceleration input proposed by Wu[2]. Both algorithms incorporate the vestibular models described in the previous section and employ identical polynomial coefficients for scaling the aircraft input. The cost function weights are kept the same for both algorithms and tuned to produce optimum responses for both pitch and surge inputs. The translational break frequency as given in Eqn. (15) was increased from 1 rad/s to 4π rad/s in both algorithms to minimize an onset false cue for responses to a surge step input.

Figure 4 compares specific force responses in the x-axis (positive sense forward) for an aircraft surge ramp to step input of 1 m/s^2 magnitude and 3 $m/s^2/s$ slope. Note that the responses are nearly identical with no onset false cue. Figure 5 compares responses to a pitch acceleration doublet input of 0.05 rad/s^2 magnitude for a 5-second duration. Note that for the angular acceleration algorithm the pulse doublet is directly input to the rotational filter W_{11}, producing a discontinuity at the doublet transition points. For the angular acceleration the pulse doublet is first integrated to a smooth triangular angular velocity which is then input to the filter W_{11}.

Objective comparison of responses to calibrated aircraft inputs for the optimal algorithm with angular velocity input are made with the adaptive algorithm. For both algorithms a time step of 0.025 seconds (an update rate of 40 Hz) was chosen to match the NASA Boeing 737-100 simulation. Polynomial scaling coefficients for each algorithm are tuned separately to optimize performance for the actuator stroke limits of the VMS. Comparisons are made of both specific force at the pilot's head and platform angular velocity, as well as vestibular responses.

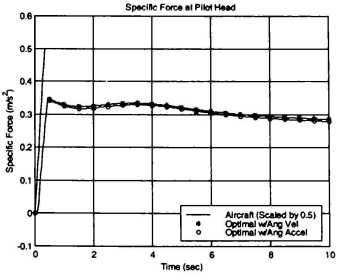


Figure 4. Comparison of Optimal Algorithm Responses to Aircraft Surge Input.

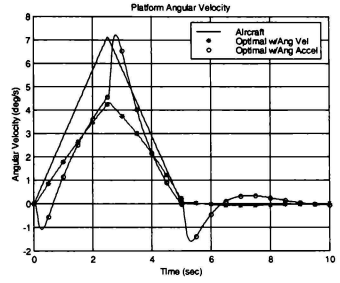


Figure 5. Comparison of Optimal Algorithm Responses to Aircraft Pitch Input.

The specific force responses to a ramp to step surge input of magnitude 1 m/s^2 and slope 3 $m/s^2/s$ are shown in Figure 6. The adaptive algorithm produces a significant false cue (-0.5 m/s^2) at onset, after which the peak is followed by a "sag" (decrease followed by increase) for about 5 seconds until a steady magnitude is reached. The optimal algorithm produces no false cue with a smooth ramp at onset followed by a smaller peak magnitude and faster washout. The sensed specific force responses show the simulator pilot response from the optimal algorithm, while reduced in magnitude, closely tracks the shape of the perceived response of the aircraft pilot. The adaptive algorithm

does not track the shape of the aircraft pilot sensed response as well, especially for the duration where the sag occurred.

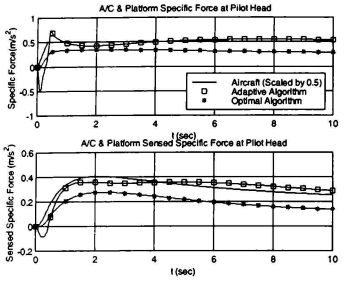


Figure 6. Comparison of Adaptive and Optimal Algorithm Responses to Aircraft Surge Input.

Angular velocity (pitch) responses due to tilt coordination generated by the surge cue are shown in Figure 7. The responses show a lower peak velocity at onset for the optimal algorithm by about 1 deg/s but followed by a negative peak of about 1 deg/s before the platform settles to zero velocity. The adaptive algorithm settles to zero velocity with no negative peak.

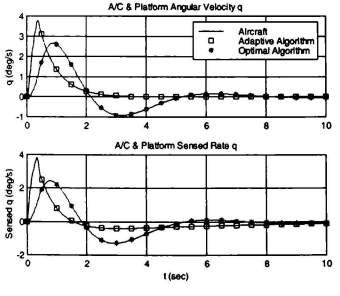


Figure 7. Platform Tilt Coordination Responses to Aircraft Surge Input.

Figure 8 shows the angular velocity responses to a pitch acceleration doublet input of 0.05 rad/s^2 magnitude for a 5-second duration. The algorithm responses are nearly identical; each response is a proportionately reduced magnitude of the aircraft angular velocity input. Figure 9 shows the specific force response in the z-axis (positive sense down) due to the pitch cue. Note that the response for the optimal algorithm is smaller in magnitude (and closer to the aircraft response) as compared to the adaptive algorithm response; this is consistent with the slightly larger pitch cue shown in Figure 7.

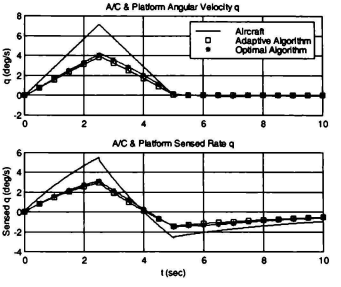


Figure 8. Comparison of Adaptive and Optimal Algorithm Responses to Aircraft Pitch Input.

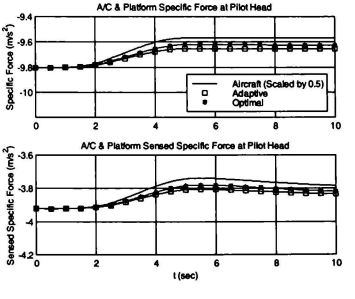


Figure 9. Z-axis Specific Force Responses to Aircraft Pitch Input.

Both the adaptive and optimal algorithms were implemented in the real time environment on the VMS and were tested with piloted simulation maneuvers. The pilot was required to execute a series of prescribed maneuvers for each algorithm. One such maneuver is a column doublet in which the pilot uses the column control input to pitch the aircraft down and then up.

Figure 10 shows the angular velocity response for the pilot maneuver executed with the optimal algorithm. For comparison the adaptive algorithm response was generated off-line from the recorded aircraft input, producing a predicted response to compare with the measured response recorded for the optimal algorithm. Note that the adaptive algorithm has an angular velocity greater than zero at the onset that results from a tilt response generated from the aircraft trim acceleration input. On the VMS the motion platform pitch angle is adjusted to the trim inputs during a "hold" mode prior to the start of a maneuver, with the platform already at rest when the maneuver is executed in "operate" mode.

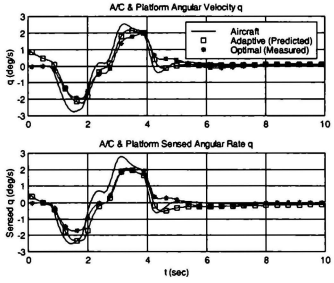


Figure 10. Algorithm Comparison of Pilot Test Column Doublet Maneuver.

As shown in Figure 10 both algorithms track the high magnitude, low frequency peaks fairly well. The adaptive algorithm does a better job at "adapting" to low magnitude, high frequency variations; especially the peak at about 2.5 seconds and the settling effect at about 4 seconds.

Figure 11 compares the specific force response in the z-axis for the column doublet maneuver. Note that the aircraft specific force starts at a value of less magnitude than the algorithm responses due to the tilt angle generated from the aircraft trim acceleration input. The aircraft input is also scaled by 10 per cent since the cueing algorithms produce a much smaller heave response due to motion platform excursion limits. Both algorithms perform about the same in tracking the large heave input peak from the aircraft, with the response washing out very quickly.

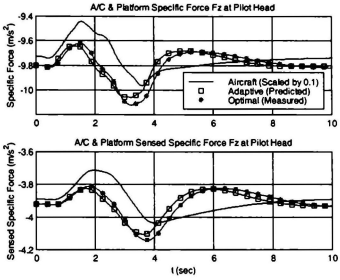


Figure 11. Z-axis Specific Force Responses to Pilot Test Column Doublet Maneuver.

Future Developments

A novel approach to motion cueing, currently being researched by the authors, is to develop a motion cueing algorithm that combines features of both the adaptive and optimal algorithms. The algorithm would be formulated as an optimal control problem with a nonlinear control law that would result in a set of adaptive cueing filters. These cueing filters can then be adjusted in real time based upon the system states; in particular those associated with perceptual errors. The control law will require the matrix Riccati equation to be solved in real time. A highly favorable approach to this computationally challenging problem is a recurrent neural network proposed by Wang and Wu[15]. The proposed algorithm will also incorporate a new otoliths model and a model of visual motion perception.

The current otoliths model given in Eqn. (5) is based upon subjective responses of test subjects. From physiological experiments, Fernandez and Goldberg developed the following transfer function for the otoliths:[16]

$$\frac{AFR(s)}{f(s)} = K\left(\frac{1+k_a\tau_a s}{1+\tau_a s}\right)\left(\frac{1+k_v(\tau_v s)^{k_v}}{1+\tau_m s}\right) \tag{26}$$

where AFR is the afferent firing rate of the vestibular neuron. Note that the numerator in Eqn. (26) contains a fractional derivative term that poses an interesting problem when implementing in state space notation in the optimal algorithm.

Young[17] noted that visual motion cues are dominant in the perception of velocity and steady state orientation at low frequencies below 0.1 Hz. At higher frequencies, vestibular cues tend to dominate. When visual and vestibular cues conflict, in particular with the direction of motion, vestibular cues will initially dominate. Motion perception can be sustained by visual cues after vestibular cues have been washed out due to motion platform limits. Visual cues introduce a bias to the perceived angular velocity in the presence of platform motion. Zacharias[18] developed functional models of how visual and vestibular cues operate in conjunction to produce human motion perception.

The current adaptive and optimal algorithms along with the new algorithm will be implemented and evaluated on a new motion system in the Cockpit Motion Facility (CMF). The CMF, as shown in Figure 12, is made up of one motion system site and four fixed-base sites. The motion system site contains a six-degree-of-freedom state-of-the-art synergistic motion base with 76-inch extension actuators. The four fixed-base sites provide homes for the simulator cockpits when they are not resident on the motion system. Each cockpit has its own visual display system and all cockpits share Evans and Sutherland ESIG 3000 or Harmony image generators.

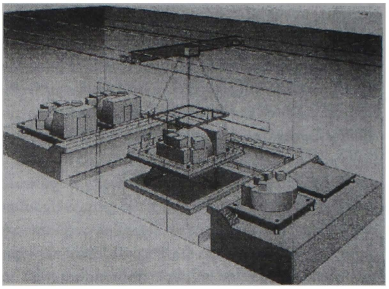


Figure 12. Cockpit Motion Facility (CMF).

The effectiveness of the proposed algorithm as compared to the current adaptive and optimal algorithms will be assessed in piloted simulations on the CMF. A series of aircraft maneuvers will be executed for each algorithm. Pilot perception (as computed from vestibular and visual motion models employing platform motion as a stimulus) and pilot control input will be recorded for each maneuver. From pilot control inputs, power spectral density, crossover frequency, and phase angle will be analyzed to determine the effect of motion platform response upon pilot performance. From these data, the fidelity of each algorithm will be benchmarked in replicating pilot performance and workload of actual aircraft maneuvers.

Conclusions

Further investigation of the optimal algorithm revealed that a revised development based upon angular velocity input is an improvement over the former approach based upon angular acceleration input. Comparisons of the angular velocity response show that distortion is eliminated with the angular velocity approach, with a cueing response very close to that of the adaptive algorithm. Pilot testing on the VMS platform confirmed these results and revealed that the adaptive algorithm is more capable of tracking small changes in aircraft angular velocity inputs. Cueing responses to a surge input show the optimal algorithm

has improved tracking capability without a false cue and lower tilt rate at onset, but producing a lower magnitude response than the adaptive algorithm.

This revised approach to the optimal algorithm will be used in the future development of an optimal algorithm that will be capable of adapting to platform motion and sensation errors in real time. This new technique will include new features not available in the current algorithms, a revised otolith model and a visual motion perception model. The new algorithm will be tested and implemented on the CMF motion platform currently being installed at the NASA Langley Research Center. Both the new motion cueing algorithm and the new motion platform facility will be instrumental in future motion studies research.

References

[1] Wu, W., and Cardullo, F. M., Is There an Optimum Cueing Algorithm?, AIAA Modeling and Simulation Technologies Conference, New Orleans, LA, August 11-13, 1997, pp. 23-29.

[2] Wu, W., Development of Cueing Algorithm for the Control of Simulator Motion Systems, MS Thesis, State University of New York at Binghamton, May 1997.

[3] Martin, D. J., Jr.; A Digital Program for Motion Washout on Langley's Six-degree-of-freedom Motion Simulator, NASA Contractor Report 145219, Contract NAS1-14600, July 1977.

[4] Sivan, R., Ish-shalom, J. and Huang J. K., An Optimal Control Approach to the Design of Moving Flight Simulators, IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-12, no.6, November/December 1982, pp. 818-827.

[5] Reid, L. D. and Nahon, M. A., Flight Simulation Motion-base Drive Algorithms: Part 1 - Developing and Testing the Equations, UTIAS Report No. 296, CN ISSN 0082-5255, December 1985.

[6] Young, L. R.; Perception of the Body in Space: Mechanisms, Handbook of Physiology – The Nervous System, Chapter 22, pp. 1023-1066.

[7] Parrish, R. V., Dieudonne, J. E., Martin, D. J., Jr., and Copeland, J. L., Compensation Based on Linearized Analysis for a Six-Degree-of-Freedom Motion Simulator, NASA TND-7349, 1973.

[8] Parrish, R. V., Dieudonne, J. E., and Martin, D. J., Jr., Motion Software for a Synergistic Six-Degree-of-Freedom Motion Base, NASA TND-7350, 1973.

[9] DeLoach, R. and Houck, J. A., Pilot Evaluation of Population-Minimal Ground Tracks in the Airport Community, AIAA Journal of Aircraft, Volume 24, Number 9, September 1987.

[10] Zacharias, G. L., Motion Cue Models for Pilot-Vehicle Analysis, Defense Documentation Center, Defense Logistics Agency, Cameron Station, Alexandria, VA, May, 1978.

[11] Goldberg, J. M. and Fernandez, C., Physiology of Peripheral Neurons Innervating Semicircular Canals of the Squirrel Monkey. I, II, III, Journal of Neurophysiology, vol. 34, no. 4, 1971, pp. 635-685.

[12] Meiry, J. L., The Vestibular System and Human Dynamic Space Orientation, NASA CR-628, October 1966.

[13] Brogan, W. L., Modern Control Theory, Third Edition, Prentice-Hall, 1991, pp. 82 – 83.

[14] Vincent, T. L., and Grantham, W. J., Nonlinear and Optimal Control Systems, John Wiley and Sons, Inc., 1997, pp. 466 – 469

[15] Wang, J. and Wu, G., A multilayer recurrent neural network for solving continuous-time algebraic Riccati equations, Neural Networks, no. 11, 1998, pp. 939-950.

[16] Fernandez, C., and Goldberg, J. M., Physiology of Peripheral Neurons Innervating Otolith Organs of the Squirrel Monkey. I, II, and III, Journal of Neurophysiology, vol. 39, no. 5, 1976, pp. 971-1009

[17] Young, L. R., Visual Vestibular Interaction, Sixth International Symposium on Biocybernetics, Control Mechanisms in Bio- and Ecosystems, International Federation of Automatic Control, Leipzig, East Germany, September 12 – 16, 1977.

[18] Zacharias, G. L., Motion Sensation Dependence on Visual and Vestibular Cues, PhD. Thesis, Massachusetts Institute of Technology, September 1977.

AIAA 99-4329

MOTION FIDELITY CRITERIA FOR ROLL-LATERAL TRANSLATIONAL TASKS

Julie Mikula*
NASA Ames Research Center
Moffett Field, California

William W.Y. Chung*
Logicon Information Systems & Services
Moffett Field, California

Duc Tran†
NASA Ames Research Center
Moffett Field, California

Abstract

A systematic effort investigating the motion cueing dependencies for coordinated roll-lateral tasks was designed for this study. Previous studies suggested a possible criterion to determine required motion fidelity. This experiment was expanded to confirm the previously suggested criteria by investigating a full range of rotational and translational motion attenuation and phase distortion. Two translational tasks were developed: (1) a helicopter making a 20-ft translation in hover, and (2) a fixed-wing jet making a 20-ft translation at a cruising speed of 250 knots. Both aircraft had desired handling qualities. Motion fidelity ratings and handling qualities ratings were collected as the subjective data. The results were consistent with and extended the previously suggested fidelity criteria for coordinated roll-lateral tasks.

Nomenclature

F_{BO} lateral stick breakout force, lb
F_{lat} lateral stick force gradient, lb/in
g gravitational acceleration, ft/sec^2
K_{lat} lateral motion washout filter gain, n.d.
K_P roll motion washout filter gain, n.d.
$L_{\delta_{lat}}$ helicopter roll control power, rad/sec^2/in
L_p helicopter roll acceleration due to roll rate, 1/sec
p helicopter roll angular rate, rad/sec
$\dot{p}$ helicopter roll angular acceleration, rad/sec^2
r_z vertical displacement between pilot abdomen and the simulator rotational center, positive downward, ft
v helicopter translational velocity, y-body axis, ft/sec
$\dot{v}$ helicopter translational acceleration, y-body axis, ft/sec^2
δ_{lat} pilot lateral stick input, in
ϕ aircraft roll attitude, rad
$\dot{\phi}$ aircraft roll attitude rate, rad/sec
ω_P roll motion washout filter frequency, rad/sec
ω_y lateral motion washout filter frequency, rad/sec
ζ_P roll motion washout filter damping ratio, n.d.
ζ_y lateral motion washout filter damping ratio, n.d.

Introduction

The required motion fidelity has been a critical issue for the simulation community as applications for ground-based flight simulations have expanded. It is known that the fidelity of the ground-based flight simulation is dependent on the simulated aircraft characteristics, tasks, and hardware dynamic performance. The interactions among these elements is the primary reason that motion tuning for the motion-based flight simulators still heavily relies on subjective adjustment.[1]

Efforts have been developed to establish motion cueing fidelity criteria. Sinacori[2] proposed a Motion Fidelity Scale (MFS), which correlated pilot opinion with the motion platform magnitude attenuation and phase distortion. His results were based on a slalom maneuver with a top speed of 60 knots, banking turns up to ± 60 degrees and normal load factors to 2 g's, and a precision hover. Jex[3] presented a subjective correlation to the motion magnitude attenuation and phase distortion based on a 15 degrees bank-and-stop fighter maneuver with a fully coordinated aircraft. He developed a criterion on the sway motion necessary to keep the spurious side force cues (from the rolling motion) small enough to prevent pilot objections. Schroeder[4] subsequently investigated the motion gain dependency of the sway motion relative to the roll motion with no washout filter applied. Chung[5] then combined the previous criteria which tied the roll and lateral motion washout filter characteristics together as shown in Figure 1.

* Member AIAA.
† Senior Member of AIAA

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner.

This study was developed to further confirm the Figure 1 criterion by examining the motion filter conditions more thoroughly, with additional pilots and tasks.

Experiment Description

Aircraft Models

Two aircraft, a helicopter and a fixed-wing jet, were developed from a generic, two degree-of-freedom model. This input was effectively roll rate command and the lateral response was fully coordinated as given by equation 1,

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} L_p & 0 & 0 \\ 0 & 0 & g \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \\ \phi \end{bmatrix} + \begin{bmatrix} L_{\delta lat} \\ 0 \\ 0 \end{bmatrix} \delta_{lat} \qquad (1)$$

The model values and cockpit stick characteristics are shown in Table 1. The control power was chosen so that both aircraft would have the same steady state roll rate per stick force input. The rotational center for both model was chosen at the pilot's abdomen.

The objective to use two different force characteristics is to determine if there are any differences in motion fidelity requirements between rotary wing tasks with a lighter stick force gradient at 1 lb/in and fixed wing tasks with a heavier stick force gradient at 2 lb/in.

Task- Helicopter

The helicopter tasks was a 20-ft lateral translation performed at a constant altitude of 25 ft as shown in Figure 2. Pilots had clear dimensional information from evenly spaced windows and ground texture. The task started in a hover and then translated 20-ft to the right, followed by 20 seconds of hover station keeping in a medium level of sum-of-sines disturbance, Table 2. This disturbance was summed directly with the pilot's control input. The desired time to complete the lateral translation was 7 seconds. The adequate completion time was 11 seconds. The desired station keeping position error was ± 2 ft, which matched the visual scene window width for easy identification. The adequate position error was ± 5 ft.

Task- Fixed-wing Jet

The fixed-wing jet used the scene shown in Figure 3. At a cruising speed of 250 Knots and an altitude of 1000 ft, the pilot was instructed to translate from the left drop tank of the leading aircraft to the drop tank at the right. In contrast with the helicopter task, pilots had a good horizon cueing reference but with less dimensional information. The same transition time performance criteria as well as the position error criteria was used as with the helicopter task. The aircraft was positioned at the same distance from the leading aircraft as the helicopter was positioned in front of the building. Note that the model did not allow a heading change, which would be small in this maneuver. The primary differences between this task and the helicopter task was the fighter-like stick force gradient and the substantially different visual cues.

Test Facilities

The roll and lateral motion axes of the Vertical Motion Simulator (VMS), Figure 4, provided ± 18 degrees of bank and 40 ft of lateral travel. The motion and visual responses were measured using the frequency response identification technique called CIFER®[7] along with an Image Dynamic Measurement System (IDMS)[8]. Figure 5 illustrates the visual measurement setup. A Gaussian white noise input was fed into the control input and the visual and motion responses were recorded for analysis. The helicopter model's roll rate response, visual response, and platform motion responses are shown in Figure 6. The visual response approximates a 60 msec pure time delay over that of the math model. The VMS's roll motion response and lateral motion response are shown in Figure 7, which shows a well matched motion system response, as recommended[9].

Test Configuration and Procedures

The motion drive algorithm was comprised of two conventional second order high-pass washout filters, as shown in Figure 8. Both had damping ratios, ζ_p and ζ_y of 0.7. A roll washout filter generated the roll motion commands, and a lateral washout filter provided the lateral motion to mitigate the leans due to the roll motion. Four roll washout filter characteristics, Figure 9, with varying roll filter frequencies, ω_y were selected to represent low, medium, and high motion fidelity as predicted and defined in Reference 5. Six lateral washout filters with variations in magnitude, K_{lat}, and phase distortion, ω_y, as shown in Figure 9, were also developed to span medium and high fidelity according to Reference 5. The low-fidelity translational motion region was not tested due to the fact that both References 3 and 4 have adequately validated that region to be unacceptable.

The gain and washout filter frequencies used for all roll and lateral configurations are shown in Tables 3 and 4. All combinations of Table 3 and Table 4 were tested. Test configurations were randomly ordered and pilots flew each configuration at least three times before giving their ratings. Annunciators inside the cockpit were used to inform pilots of their tasks performance for both the translational time and station-keeping position error. Five experienced pilots, which included two Navy Test Pilot School instructors, two NASA test pilots, and one retired NASA test pilot, participated in this investigation. Pilots were asked to give handling qualities ratings (HQRs)[6] and motion fidelity scale (MFS) ratings as shown in Table 5.

Results

Test data were analyzed according to the roll motion configurations as shown in Table 3, and in Figure 9. The first group, A1, was analyzed for full roll motion (A1), i.e., the simulator bank angle followed the helicopter bank angle without any magnitude attenuation and phase distortion. For this condition, the variables were: lateral gain (K_P) and lateral phase distortion by changing the washout filter frequency (ω_P).

The second group, was analyzed for the attenuated roll motion with magnitude gain of 0.6 and three levels of phase distortion, i.e., phase distortion at 1 rad/sec of 20 degrees, 43 degrees, and 80 degrees, or test points A2, A3, and A4 from Figure 9 respectively. For this roll motion gain, the vaiables were roll motion phase distortion, (by changing ω_P), lateral motion phase distortion (by changing ω_y), and lateral motion gain (K_{lat}).

The observed levels of significance (p-values) were determined for these two groups of data, as shown in Table 6. A p-value is the probability of being incorrect in stating that an experimental factor (such as translational phase distortion) is causing the variation of the data (such as the handling qualities rating) rather than the variation being due to random effects. Typically, when the p-value is less than 0.05 (5 chances in 100 of making an error), the results are deemed statistically significant.

Helicopter - Full roll motion (A1):

The results from Table 6 show there is a significant effect due to the lateral motion gain on the average MFS (p=0.037) and HQR (p=0.042). The average MFS degraded from 1.7 to 1.3, and HQR degraded from 5.2 to 5.6 as the lateral motion gain (K_{lat}) was reduced from 0.8 to 0.5.

The data also show there is a significant effect due to the combination of the lateral motion gain and lateral washout filter frequency (lateral phase distortion) on the average MFS (p=0.004) and HQR (p=0.023). Figure 10 shows the average motion fidelity scale rating degraded as the lateral washout filter frequency (ω_y), increased from 0.25 to 0.9 with the interaction from the lateral motion gain.

The results suggest when the roll motion cues match the visual cues, there is a benefit to minimize the lateral phase distortion and use as much lateral motion gain to mitigate the leans due to the roll motion. However, when lateral phase distortion is high ($\omega_P \geq 0.5$), little benefit is gained with using larger lateral motion gain. This is consistent with the Reference 5 findings.

Helicopter - Attenuated roll motion (A2, A3, A4):

There were significant motion fidelity dependencies found due to the roll motion phase distortion (p=0.026), and the lateral motion phase distortion (p=0.022). Figure 11 shows the averaged MFS degraded as the roll motion phase distortion (ω_P) increased. Figure 12 shows the averaged motion fidelity scales degraded as the lateral washout filter frequency (ω_y) increased from 0.25 to 0.9.

From Table 6, the handling qualities was found to be affected by the lateral motion phase distortion (p=0.022) only. Figure 12 shows the average HQR degraded as the lateral washout filter frequency increased from 0.25 to 0.9.

There is also a marginal motion fidelity dependency ($0.1>p>0.05$) found due to the lateral motion gain (p=0.072) and the combination of the roll phase distortion and the lateral gain (p=0.091). As shown in Figure 13, the average motion fidelity scales improved when the roll motion phase distortion is relatively small (ω_P=0.25 and 0.5) when K_{lat} =0.8. However, for the largest roll phase distortion (ω_P=0.9), the larger lateral motion gain had a negative effect on the motion fidelity. The rationale may lie in the distorted roll motion cues were simply reinforced by the erroneous side force cues.

The results suggest that motion fidelity is dependent on the roll motion phase distortion, the lateral motion phase distortion, and the lateral motion gain. The results suggest there is a benefit to keep both the roll motion phase distortion and the lateral motion phase distortion as low as possible, and provide as much lateral motion gain as possible when low phase distortions are applied. This is consistent with Reference 5 findings.

By defining high motion fidelity as the mean-MFS≥2.5, medium motion fidelity as the mean-MFS between 2.5 and 1.5, and the low motion fidelity as the mean-MFS less than 1.5, the average MFS from the helicopter task are compared with the criteria proposed in Reference 5 as shown in Figure 14. The results show good match in the expected high and medium motion fidelity regions. In the expected low motion fidelity region, only 4 out of 10 test outcomes match the predicted fidelity. For those unmatched low motion fidelity cases, the trend of degradation and the average MFS are still leaning toward the low motion fidelity.

Fixed-Wing

Due to limited time available, only three roll motion configurations and four lateral motion configurations were evaluated. The three roll motion configurations were A1, A2, and A3; and the four lateral motion configurations were T1, T4, T5, and T6 as shown in Figure 9.

Fixed Wing - Full roll motion (A1):

The average motion fidelity degraded as the lateral motion gain decreased from 0.8 to 0.5 as shown in Figure 15. The average motion fidelity degraded as the lateral phase distortion increased as shown in Figure 16. These results are consistent with the helicopter task. The average

HQR shows the same consistency as shown in the average HQR graphs in Figure 15 and 16.

Fixed-Wing - Attenuated roll motion (A2, A3):

Both the averaged MFS and HQR degraded as the roll motion phase distortion increased as shown in Figure 17. Both the averaged MFS and HQR also degraded as the lateral motion phase distortion increases. Figure 18 shows the MFS and HQR degrading as the lateral washout frequency (ω_y) increased which is also consistent with the helicopter task results..

In general, results from this evaluation are quite consistent with the helicopter hover task. This is significant in establishing that even with different control power settings (0.67 vs. 1.33 rad/sec2/in.), different stick force gradients (1.0 vs. 2.0 lb/in.) and different tasks and visuals scenes the results were consistent.

Conclusions

1) When using roll and lateral motion filters in a coordinated task, the roll phase distortion, lateral phase distortion, and lateral gain were shown to have effects to the motion fidelity. The findings suggest there is a benefit to keep the phase distortion, i.e., the washout filter frequency, of both washout filters small to improve the motion fidelity. In addition, the findings show there is a benefit to keep the lateral gain large when the phase distortion is small to improve the motion fidelity

2). The findings support the fidelity criteria proposed in Reference 5.

References

[1]Grant, P.R. and Reid, L.D.: "Motion Washout Filter Tuning: Rules and Requirements," AIAA Flight Simulation Technologies Conference, August 1995.

[2]Sinacori, J. B.: "The Determination of Some Requirements for a Helicopter Flight Research Simulation Facility," NASA CR 152066, September 1977.

[3]Jex, H.R., Jewell, W.F., Magdaleno, R.E., and Junker, A.M., "Effects of Various Lateral-Beam Washouts on Pilot Tracking and Opinion in the Lamar Simulator," AFFDL-TR-79-3134, pp. 244-266.

[4]Schroeder, J.A. and Chung, W.Y.: "Effects of Roll and Lateral Flight Simulation Motion Gains on a Sidestep Task," American Helicopter Society's 53rd Annual Forum, April 1997.

[5]Chung, W.Y., Robinson, D.J., Wong, J., and Tran, D.: "Investigation of Roll-Lateral Coordinated Motion Requirements with a Conventional Hexapod Motion Platform," AIAA Motion and Simulation Techonologies Conference, AIAA-98-4172, August 1998.

[6]Cooper, G. E.; and Harper, R. P., Jr.: "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities," NASA TN D-5153, April 1969.

[7]Tischler, M. B., Cauffman, M.G.: "Frequency-Response Method for Rotorcraft System Identification: Flight Applications to BO-105 Coupled Rotor/Fuselage Dynamics," Journal of the American Helicopter Society, Vol 37, No 3, pgs 3-17, July 1992.

[8]Lehmer, R.D., and Chung, W.Y.: "Image Dynamic Measurement System (IDMS-2) for Flight Simulation Fidelity Verification," AIAA Modeling and Simulation Conference, AIAA 99-4035, August 1999.

[9]Chung, W.Y. and Schroeder, J.A.: "Visual and Roll-Lateral Motion Cueing Synchronization Requirements for Motion-Based Flight Simulations," American Helicopter Society's 53rd Annual Forum, April 1997.

Table 1. Simulated aircraft response and force characteristics

Aircraft	L_p	$L_{\delta_{lat}}$	F_{lat}	F_{BO}
Helicopter	- 4	0.67	1.0	0.5
Fixed Wing	- 4	1.33	2.0	0.5

Table 2. External disturbance

Frequency (rad/sec)	0.28	0.49	0.8	1.5	2.67	4.6	8.5
Amplitude (in)	0.002	0.006	0.014	0.032	0.054	0.068	0.06

Angular configuration	Motion Gain, K_p	Washout filter Frequency, ω_p (rad/sec)	@ 1 rad/sec Gain	@ 1 rad/sec Phase distortion (deg)
A1	1.0	0.0001	1	0.0
A2	0.6	0.25	0.6	20
A3	0.6	0.5	0.58	43
A4	0.6	0.9	0.47	80

Table 4. Coordinated lateral motion washout filter configurations

Coordinated translation configuration	Motion Gain, K_{Lat}	Washout filter Frequency, ω_y (rad/sec)	@ 1 rad/sec Gain	@ 1 rad/sec Phase distortion relative to angular motion (deg)
T1	0.8	0.25	0.8	20
T2	0.8	0.5	0.78	43
T3	0.8	0.9	0.63	80
T4	0.5	0.25	0.5	20
T5	0.5	0.5	0.49	43
T6	0.5	0.9	0.4	80

Table 5. Motion fidelity scale

	Description	Score
High Fidelity	Motion sensations are not noticeably different from those of visual flight	3
Medium Fidelity	Motion sensations are noticeably different from those of visual flight, but not objectionable	2
Low Fidelity	Motion sensations are noticeably different from those of visual flight and objectionable	1

Table 6. Observed levels of significance (p)

Full Roll Motion	Motion Fidelity Scale	Handling Qualities Rating
Lateral gain	0.037	0.042
Lateral phase distortion	0.067*	no significance
Lateral gain & lateral phase distortion	0.004	0.023
Attenuated Roll Motion		
Roll phase distortion	0.026	no significance
Lateral phase distortion	0.022	0.022
Lateral gain	0.072*	no significance
Roll phase distortion & lateral gain	0.091*	no significance

*Marginally significant

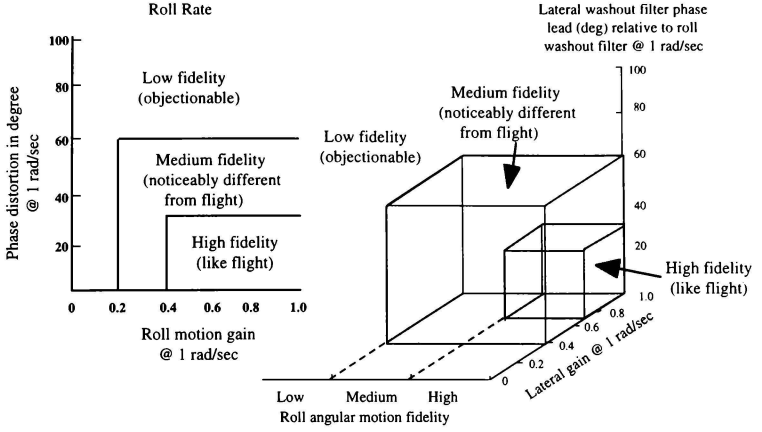


Figure 1. Motion fidelity criteria for the roll-lateral task

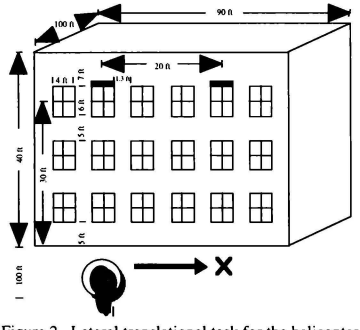


Figure 2. Lateral translational task for the helicopter

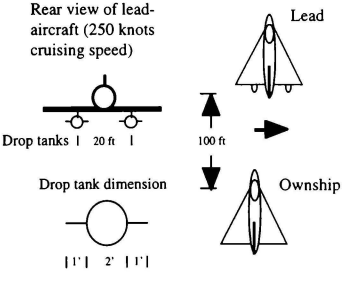


Figure 3. Formation flight task for the fixed-sing jet

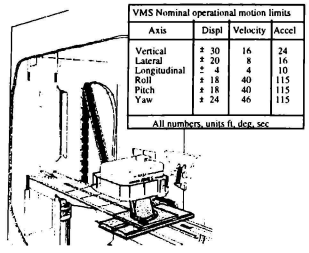


Figure 4. Vertical Motion Simulator (VMS)

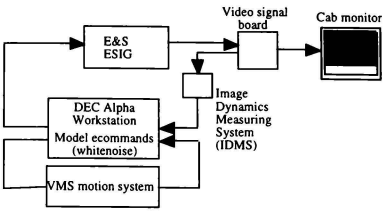


Figure 5. Visual delay measurement setup by white noise or frequency sweep

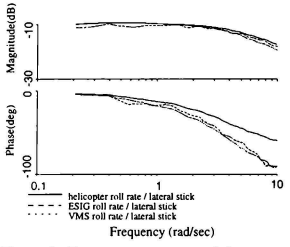


Figure 6. Frequency response of the test model roll rate response, visual throughput response, and the roll motion response

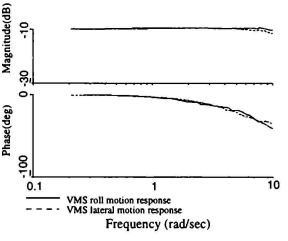


Figure 7. Frequency response of the VMS roll motion response vs. the VMS lateral motion response

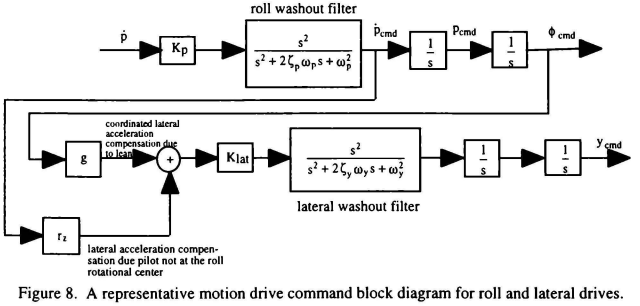
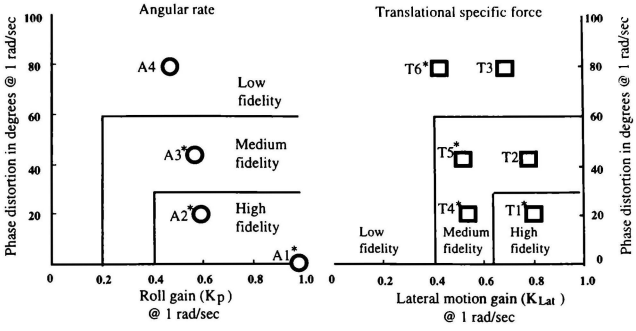
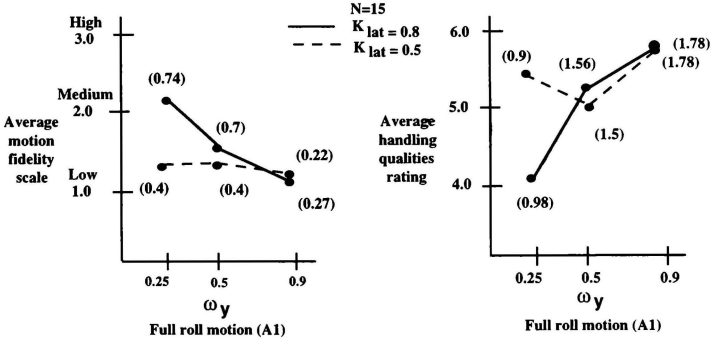


Figure 8. A representative motion drive command block diagram for roll and lateral drives.



* Also tested for the fixed-wing task

Figure 9. Test motion washout filter configurations for the helicopter translational task



N: number of samples
(*) standard deviation

Figure 10 Combined effects due to lateral gain and phase distrortion on the average MFS and HQR for the full motion helicopter task

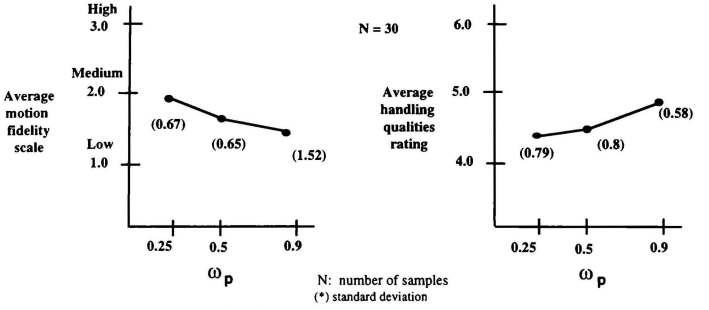


Figure 11. Roll washout filter frequency effect on the average MFS and HQR for the attenuated roll motion in the helicoper task

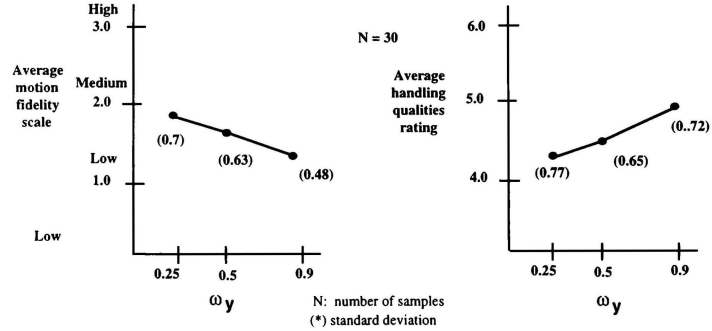


Figure 12. Lateral washout filter frequency effect on the average MFS and HQR for the attenuated roll motion in the helicoper task

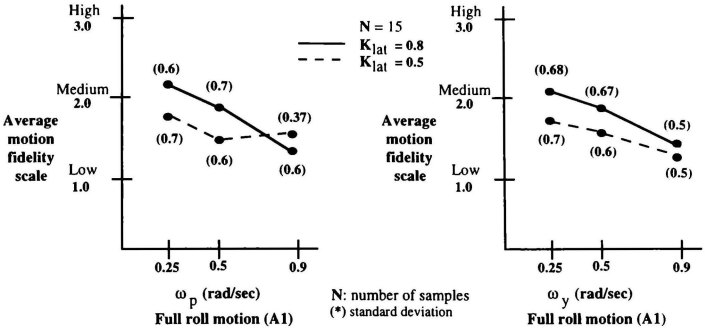


Figure 13. Lateral motion gain effect on the average MFS for the attenuated roll motionin the helicopter task

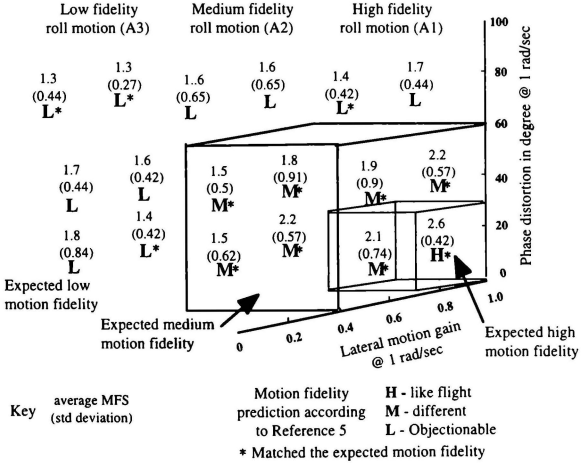


Figure 14. Comparison of the average MFS from the helicopter task with Reference 5

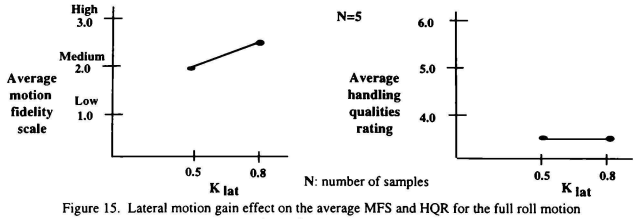


Figure 15. Lateral motion gain effect on the average MFS and HQR for the full roll motion in the fixed wing task

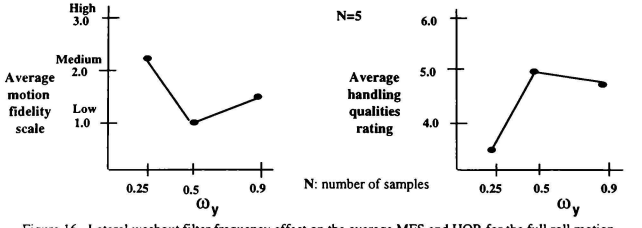


Figure 16. Lateral washout filter frequency effect on the average MFS and HQR for the full roll motion in the fixed wing task

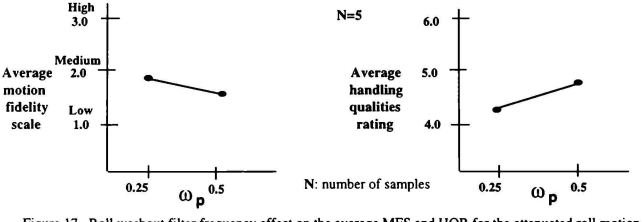


Figure 17. Roll washout filter frequency effect on the average MFS and HQR for the attenuated roll motion in the fixed wing task

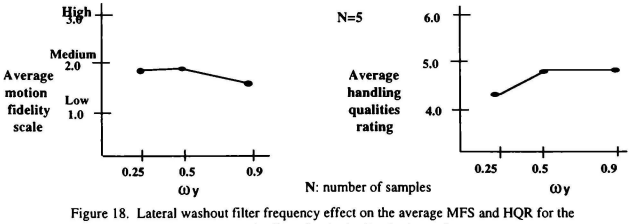


Figure 18. Lateral washout filter frequency effect on the average MFS and HQR for the attenuated roll motion in the fixed wing task

MOTION FIDELITY CRITERIA BASED ON HUMAN PERCEPTION AND PERFORMANCE

Alan D.White*
Defence Evaluation and Research Agency (DERA)
Bedford, United Kingdom

Victor V.Rodchenko†
Central Aerohydrodynamic Institute (TsAGI)
Zhukovsky, Russia

Abstract

Based on experience accumulated by DERA and TsAGI in developing and using moving-base flight simulators and on the results of recent experiments the following aspects of motion cueing are considered: the role of accelerations in piloting, motion system drive law criteria and the requirements for motion system dynamic performance. The paper presents a theoretical approach to estimating the effect of accelerations, proposes criteria to evaluate motion fidelity for different driving laws and identifies the requirements for motion platform dynamics. Estimates according to the theoretical approach and criteria are compared with the experimental data.

Introduction

The continuing improvement in computer-assisted technologies permits high-fidelity simulation of aircraft dynamics, manipulator forces, visual and sound cues. However, the fidelity of motion cueing is less dependent on computer technologies and, in addition, motion cues can never be reproduced exactly on ground-based simulators. For these and other reasons motion simulation fidelity is a key factor in determining the potential of modern flight simulators and the validity of simulation results.

In recent years motion fidelity has been improved by refining the technical means of acceleration reproduction[1,2]. Simulator displacement capabilities have been increased and motion system dynamic performance improved[1,2]. New motion cueing devices have been developed: dynamic seats, g-suits and others[1]. In some respects, the technical characteristics of these devices are approaching practical limits. For example, the displacement capabilities of the largest research simulators are several metres[1,3,4]. It is unlikely that a platform with much larger displacement could be created for less than the cost of an aircraft. Nevertheless, some distortions in the reproduction of aircraft accelerations are inevitable, however sophisticated the simulator may be, and thus motion fidelity may be less than desired. The expense and constraints associated with motion platforms are usually significant but it is not yet clear how well the simulated motion cues need to match the real cues. Moreover, distortions in the reproduction of motion cues may even have a more adverse effect than not having motion cues at all. Thus, the motion cueing problem becomes more scientific than technical. Among the main scientific issues of motion cueing are the following: (1) how do motion cues affect control of an aircraft? (2) how can motion system drive algorithms be improved? (3) what requirements must the motion devices meet?

Usually these issues are addressed experimentally. However, experimental data alone is insufficient to solve these problems because, first, the whole spectrum of cueing device characteristics and modeled flight conditions cannot be considered in experiments and, second, reliable data are difficult to obtain in experiments. The point here is that, on the one hand, simulator displacement limitations do not allow exact motion cue reproduction for most flight conditions simulated; on the other hand, real flight does not allow motion cues to be "switched off" or "distorted", while other flight conditions remain

* Principal Scientist, Flight Management and Control Department
† Head of Section, Flight Dynamics & Control System Department

© British Crown copyright 1999. Published with the permission of DERA on behalf of the Controller of HMSO.
Copyright © 1999 by TsAGI. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

unchanged. Reliable results can be obtained by comparing ground-based and in-flight simulation data but such comparative investigations are expensive and, therefore, comparatively rare. To successfully solve motion cueing problems a theoretical approach must be developed. However, no theoretical methods which would encompass the wide variety of available experimental data have been developed to date.

This paper presents a theoretical approach to estimating the effects of acceleration on piloting, criteria for the selection of drive laws and requirements for motion platform dynamic performance.

The Role of Motion Cues in Piloting

It is well known that high g-loads (n_z>5) or persistent changes in the level of acceleration (Δn_z>0.3-0.5) can have a considerable effect on manual control performance through changes in a pilot's physiological state: increase in pulse, breathing rate and blood pressure, tiredness, etc. Less obvious is the effect of small accelerations typical of common flight missions for all types of aircraft and helicopters (take-off, landing approach, etc.).

Accelerations of this type do not usually affect the psycho-physiological state of a pilot, but they may have a significant effect on piloting, as can be seen, for example, in fig.1 (from reference 5). This compares the Handling Qualities Levels from real flight with fixed-base simulations for different roll mode time constant τ_R and lateral control sensitivity value $L_{F\delta s}$. There is a world of difference between the data obtained in ground-based experiments and in flight - Level 1 and Level 3 seem to switch places in the figure.

The effect of accelerations on piloting depends in a complicated way on the aircraft characteristics, the control axis and other factors. An understanding of the underlying mechanisms would be helpful in solving many other motion cueing problems, such as estimating the need for motion cueing in particular cases or correcting pilot ratings obtained on a fixed-base simulator. Existing theoretical approaches to aircraft controllability cannot explain some acceleration effects even qualitatively. For example, according to an existing theory[6] the best handling qualities are achieved at τ_R=0, which is supported by fixed-base simulatio results (fig.1 upper). Contrary to this theory, in real flight (fig.1 middle) handling qualities for $\tau_R < 0.5$ *sec* are degraded due to the acceleration effect.

Recently TsAGI developed a theoretical approach to the analysis of acceleration effects. In this approach acceleration effects are represented by differences (ΔPR) between the moving-base (PR_m) and fixed-base (PR_f) Cooper-Harper pilot ratings.

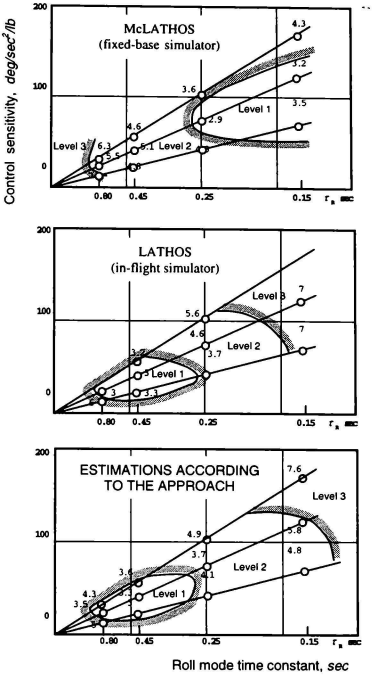


Fig.1 - The comparison of fixed-base and flight test data with the estimations according to the approach proposed.

It can be seen from fig.1 that accelerations can play both beneficial and negative roles in piloting. Thus ΔPR can be presented as follows:

$$\Delta PR = \Delta PR^{-} - \Delta PR^{+}$$

where ΔPR^{-} and ΔPR^{+} are pilot rating increments due to negative and beneficial effects respectively.

The main principles of the approach are applicable to all control axes. In this paper the lateral control axis is used to demonstrate the effectiveness of the approach as there are sufficient experimental data for this control axis.

Beneficial acceleration effect. The beneficial effect of motion cues on lateral control is accounted for by the fact that motion cues lead roll visual cues and, unlike visual cues, motion cues are perceived regardless of pilot attention. Motion cues promote a reduction in the pilot's pure time delay which increases pilot-aircraft system stability, widens bandwidth, and allows a pilot to increase his gain. As a result the quality of control is better and pilot ratings improve[7,8].

Bandwidth is a useful parameter that enables an assessment to be made of aircraft controllability[9]. As the beneficial acceleration effect is connected with the variation in pilot-aircraft system time delay and the need for pilot lead compensation, it would be natural to suppose that this effect depends on the bandwidth value ω_{BW}: the greater ω_{BW}, the less the influence of time delay and phase lead on the beneficial effect, and vice versa. Analysis shows that good agreement between estimations and experimental data is achieved if the beneficial effect is determined by the following expression:

$$\Delta PR^{+} = \begin{cases} 2, \omega_{BW} < 1.5 rad/sec \\ 2.55 - 3.14 \cdot \log \omega_{BW}, \ 1.5 \le \omega_{BW} < 6 \\ 0, \omega_{BW} \ge 6 rad/sec \end{cases}$$

The bandwidth ω_{BW} is determined from the condition :

$\varphi\left[Y_{\left(p/\delta_{as}\right)}(j\omega_{BW})\right] = -45^{o}$, where p is roll rate and δ_{as} is aileron stick displacement.

Negative acceleration effect. It is well known that some aircraft characteristics (for example, short roll mode time constants, see fig.1) may cause abrupt aircraft responses to pilot control inputs. This phenomenon is accounted for by an aircraft's response to the high-frequency component of a pilot's control input. As a result high-frequency angular accelerations arise, which cause significant high-frequency (0.5-3 Hz) linear accelerations, depending on the height of the pilot's position relative to the rotational axis. These accelerations not only cause unpleasant sensations but also hamper the pilot's perception of "useful" information about angular motion.

A pilot's sensitivity to such lateral accelerations depends on the roll rate amplitude[10]. For roll amplitudes greater than 1 deg/sec the intensity of lateral acceleration sensations is determined by the ratio of the root-mean-square of lateral accelerations σ_{ny} and roll rates σ_p, i.e., $\lambda = \frac{\sigma_{ny}}{\sigma_p}$. In this approach the negative acceleration effect is estimated by parameter λ according to the function:

$$\Delta PR^{-} = \begin{cases} 0, \ \lambda < 0.2 \\ 9 \cdot \log\lambda + 6.3, \ \lambda \ge 0.2 \end{cases}$$

The value of λ can be calculated if we assume that a pilot's stick deflections can be represented by white noise passing through a filter of the following type:

$$Y_{pilot} = \frac{1}{s + \omega_c \cdot K_p / K_{p*}},$$

where $K_p = Y_{\left(p/\delta_{as}\right)}(j\omega)|_{\omega=0}$ i.e. the steady-state gain in the stick to roll rate transfer function $Y_{\left(p/\delta_{as}\right)}(j\omega)$, K_{p*}=0.025 *rad/sec/mm* and ω_c = 2 *rad/sec* i.e. the pilot-aircraft system cut off frequency.

Then, according to the well-known expressions from random function theory, the values of σ_{ny} and σ_p, can be determined by the following expressions:

$$\sigma_{ny} = \frac{h}{g} \cdot \frac{1}{2\pi} \cdot \int_{-\infty}^{+\infty} \omega^2 \left| Y_{\left(p/\delta_{as}\right)}(j\omega) \cdot Y_{pilot}(j\omega) \right|^2 d\omega$$

$$\sigma_p = \frac{1}{2\pi} \cdot \int_{-\infty}^{+\infty} \left| Y_{\left(p/\delta_{as}\right)}(j\omega) \cdot Y_{pilot}(j\omega) \right|^2 d\omega$$

where h is the height of pilot's position relative to the rotational axis.

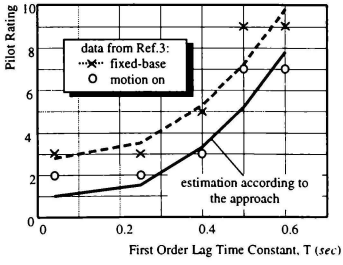


Fig.2 - Effects of accelerations for different prefilter time constants (τ_k=0.5984 *sec*).

The proposed approach produces results which agree with various flight and ground-based test results

for conventional aircraft, helicopters and VTOL. Good correlation between the estimations and experimental data is demonstrated for example in fig.1 for different roll mode time constants and control sensitivities, and in fig.2 for different prefilter time constants.

The effects of accelerations on longitudinal control for aircraft of traditional configuration differ markedly from the effects on lateral control. Fig.3 shows the experimental data from one fixed-base simulation and two moving-base tests, different degrees of freedom on, i.e. pitch only, pitch and heave simultaneously. It is obvious that the pitch acceleration effect depends on whether normal accelerations are present. Pitch motion affects piloting beneficially but simultaneous modelling of the normal and angular accelerations (as in real flight) leads to the beneficial effect disappearing. It follows from these data that flight simulation with pitch motion only may lead to misleading results.

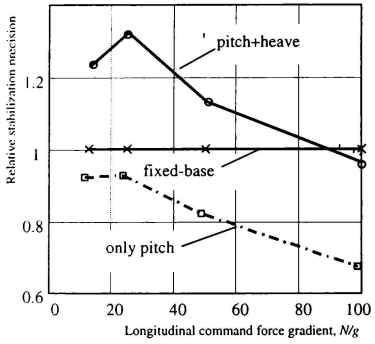


Fig.3 - The influence of different motion cues on pitch piloting precision.

For a traditional configuration aircraft there is an inseparable link between load factor and pitch attitude:

$$\frac{n_z}{\theta} = \frac{n_{z_\alpha} s}{s + n_{z_\alpha} \cdot g/V}$$

This means that pitch motions are always accompanied by normal specific forces. Relative to their respective perception thresholds, the values of the specific forces considerably exceed the values of pitch accelerations. Thus the normal acceleration effect hampers pitch acceleration perception and pitch cueing does not affect piloting beneficially.

From this expression it also follows that normal accelerations within the pilot activity frequency range do not lead the pitch cues, i.e. heave motion cues do not lead visual pitch cues. Thus normal specific forces do not affect pitch control beneficially.

The approach adopted for the lateral control axis is also applicable to the longitudinal control axis. However, the effects of normal acceleration on longitudinal control must be considered in order to develop a method similar to that developed for the lateral axis.

Flight Simulator Drive Laws

The three main techniques used in motion platform drive laws are: 1) washout, or elimination of low-frequency aircraft motion components; 2) acceleration scaling, or reduction of computed accelerations through multiplying by a gain less than one; 3) cockpit tilting to imitate low-frequency lateral and longitudinal specific forces.

Although these techniques are widely used, the effect of drive law parameters on motion fidelity is still not fully understood. As a result, we do not know how to improve cueing or how to combine cues from one device with another. Nor do we understand the benefits and limitations of various cueing devices. At present the drive laws for a motion cueing device are determined principally by trial and error or based on available experimental results. This approach demands a great number of experiments. Moreover, the empirical approach cannot be applied to studying the controllability characteristics of a new aircraft, as in this case the evaluating pilot has no reference model for comparison. For these reasons motion cueing devices are probably used less effectively than they could be.

There are few criteria in existence for estimating drive law parameters. Among the best known is Sinacori's criterion[11,4]. This criterion combines washout and scaling: motion fidelity is determined by gain and phase at a frequency of 1 rad/sec. However, this criterion is based on a limited amount of experimental data for helicopters only and does not seem to apply to conventional aircraft. For example, according to the criterion[4], a scale factor (m) less then 0.4 for the rotational axes and less than 0.65 for the linear axes will provide inadequate simulation fidelity. However, pilot ratings for a fixed-wing aircraft offset landing approach task [3] on a simulator with lower scale factors (m = 0.2 in the linear axes, m = 0.3 in roll and yaw, m = 0.6 in pitch) were practically the same as those in flight, i.e. motion simulation fidelity in this case was sufficiently high.

As an alternative to the unified approach above, this paper presents separate criteria based on scale factor and washout frequency. Such an approach is justified by differences in the nature of the distortions caused by each particular motion cueing method. For example, for scaling it is acceleration attenuation whilst for high-pass filtering it is mainly phase distortions. Thus, the influence of scaling on motion fidelity does not depend on high-pass filter parameters and vice versa.

The available experimental data show that the effect of drive algorithm coefficients on motion fidelity depends on the role of motion cues in a particular case: it can be beneficial or negative. Thus, development of criteria can be simplified if we treat separately not only scale factor and washout frequency but also the different roles of motion cues in piloting (beneficial or negative).

Washout Filters. High-pass filtering is one of the most efficient methods of constraining accelerations. This method eliminates the low-frequency acceleration components that cause large cockpit displacements and do not affect aircraft control (at least, their effect is much weaker than the effect of the high-frequency components).

Washout filters of first order up to third order are normally used:

$$Y_1 = \frac{s}{s+\omega_1}\ ,\ Y_2 = \frac{s^2}{s^2+2\zeta_2\omega_2 s+\omega_2^2},\quad Y_3 = Y_1 \cdot Y_2\ .$$

The damping ratio (ζ) is usually equal to 0.7 [2,12]. Thus, to study the effect of washout on motion fidelity, we need to know only the effect of the filters' natural frequencies ω_1, ω_2,.

Fig.4 shows experimental data for the case where angular acceleration is beneficial. The figure shows relative roll stabilization precision against break frequencies of filters of different orders Y_1, Y_2, Y_3. Filter break frequency was defined in accordance with the formula:

$|Y(j\omega_{br})| = 0.7$.

It can be seen that the motion fidelity is practically the same, provided the filters' break frequencies are equal. This means that break frequency can be used as the basis for a criterion to evaluate simulation fidelity for different filters. There might be alternatives but this parameter has very clear physical meaning, it is well known and simple, and it is sufficient to evaluate motion fidelity.

Qualitatively, the functions in fig.4 also apply to linear accelerations. These and other experimental data show that, for angular degrees of freedom, high motion fidelity is achieved for break frequencies less than 0.7-1 rad/sec whilst for linear axes the boundary for high fidelity is 0.5 rad/sec. For frequencies over 2 *rad/sec* for the linear axes and over 4 *rad/sec* for the rotational axes the motion fidelity is low and comparable to fixed-base simulation.

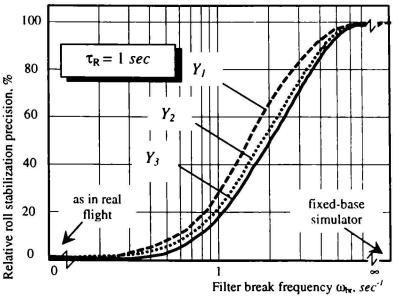


Fig.4 - Effect of washout filters of different orders on angular motion simulation fidelity where the acceleration effect is beneficial.

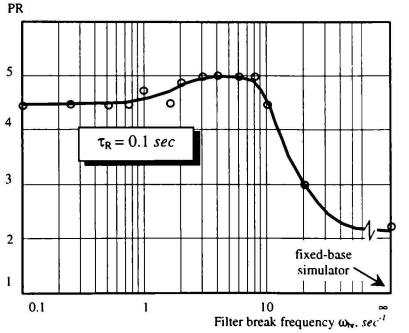


Fig.5 - The influence of washout filter break frequency on motion fidelity for negative acceleration effect.

The influence of washout filtering for cases where the effect of motion cues is negative is presented in fig.5. It can be seen that the influence of break frequency in this case is different to that for the beneficial effect. Motion fidelity starts to degrade

quickly (pilot ratings improve) for break frequencies above 5 rad/sec, instead of 0.5-1 rad/sec for the beneficial effect.

Scaling. Let us first consider the case where accelerations are beneficial. Our investigations showed that the effect of scaling is qualitatively similar for different aircraft characteristics and flight conditions. Scaling down does not have any noticeable effect on piloting if simulator angular rates (or specific forces) are considerably higher than the pilot's sensitivity threshold values. Motion fidelity starts to deteriorate once the simulator angular rates (or specific forces) fall below a certain value, which is twice the pilot's sensitivity threshold value. When simulator angular rates (or specific forces) are close to their threshold values the motion fidelity is equal to that of a fixed-base simulator.

The proposed criterion, applicable to different control axes and modeled conditions, is based on the parameter μ, defined as the ratio of simulator angular rates $m\sigma_p$ or specific forces $m\sigma_{n_z}$ to their threshold values $\sigma_{p_{th}}$, $\sigma_{n_z th}$:

$$\mu = m\frac{\sigma_p}{\sigma_{p_{th}}}, \quad \mu = m\frac{\sigma_{n_z}}{\sigma_{n_z th}} .$$

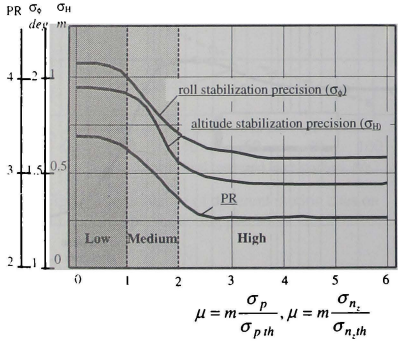


Fig.6 - A criterion to estimate the influence of scaling for beneficial acceleration effect.

Motion fidelity depends on the parameter μ according to the function in fig.6. If these ratios are less than unity, we have a low level of motion fidelity, i.e. comparable to a fixed-base simulator; if they are greater than 2, we have a high fidelity level, i.e. comparable to real flight.

The influence of scaling on motion fidelity for cases where the effect of accelerations is negative can be estimated from the function in fig.7. In these cases, unlike the beneficial effect, scaling down improves pilot ratings compared with real flight due to the lower intensity of high-frequency specific forces. For values of simulator specific force equal to or less than their threshold values, pilot ratings are unchanged and the motion fidelity is the same as for a fixed-base simulator.

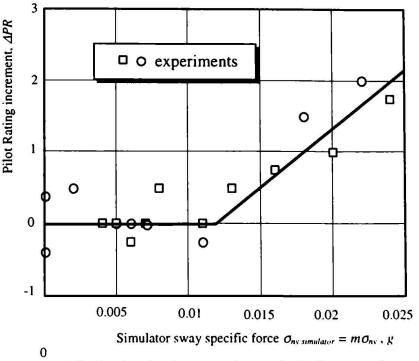


Fig.7 - A criterion to estimate the influence of scaling for negative acceleration effect.

Motion Fidelity Criteria for Tilting. Cockpit tilting is used to simulate low-frequency longitudinal and lateral specific forces. The following algorithm is usually used to tilt the cockpit (for pitch, for example):

$$\theta_t = \frac{\omega_{lp}^2}{s^2 + 2\zeta\omega_{lp}s + \omega_{lp}^2} \cdot n_x$$

The data in fig.8 demonstrate the influence of the filter's natural frequency ω_{lp} on perceived longitudinal acceleration fidelity during a take-off run. These and other data show that the principal constraints on using cockpit tilt in this way are due to false rotational sensations and false specific forces caused by jerky angular accelerations. In the area below the curves (fig.8) a pilot perceives cockpit tilting as linear accelerations. In the area above the curves, false sensations arise. To avoid these sensations the following criteria must be met: tilt velocity should not exceed approximately 2 deg/sec; false specific forces

caused by angular accelerations should not exceed 0.01g.

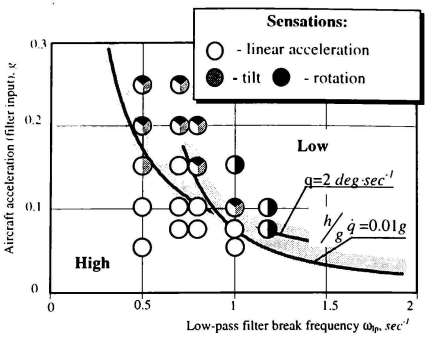


Fig.8 - The criterion for motion fidelity estimation in tilting.

Motion Platform Dynamic Performance Requirements

It is obvious that distortions in motion cue reproduction can degrade simulation fidelity. However, too rigid a requirement for the dynamic characteristics will result in a more expensive system which is why the existing Standards and available publications[13,14] pay so much attention to the dynamic characteristics of motion systems. However, the effect of these dynamic characteristics on flight simulation fidelity has not been fully researched and, thus, the standards and requirements have not been fully substantiated so far.

Acceleration distortions can be divided into two types: nonlinear distortions and describing function (amplitude and phase) distortions.

Nonlinear Distortions. These are high-frequency and felt as jerks and general roughness. As a rule, a pilot can distinguish them from the accelerations inherent in piloting itself.

To evaluate the permissible nonlinear distortions a special scale was created. The scale has four grades, *MR*=1 being the highest fidelity, when the distortions are not felt by a pilot, and *MR*=4 being the lowest fidelity, when the intensity of the jerks is such that aircraft motion cannot be distinguished due to the distortions.

The data gathered show that the permissible values of nonlinear distortion are approximately the same for all three linear degrees of freedom and do not depend on the distortion frequencies. The effect of distortions on simulation fidelity depends on the simulated acceleration amplitude. The permissible distortion values for different amplitudes of sinusoidal accelerations are shown in fig.9 (the frequency of the input accelerations is 0.5Hz, as recommended in AGARD-AR-144[13]). As used here, distortion is the maximum difference between the output accelerations and its first harmonic. From fig.9 it can be seen that the permissible distortions are determined by the absolute errors in acceleration reproduction if the acceleration amplitude is less than 0.015*g*. If the acceleration values exceed 0.015g, the permissible values of distortion increase and the motion fidelity is determined by relative errors in acceleration reproduction. To achieve a high level of motion fidelity the permissible values of relative distortions should be lower then 20% and for medium fidelity they should be within 20-50%. Relative distortions higher than 50% correspond to a low level of motion fidelity.

The approach to estimation of permissible angular motion distortions differs from that of specific forces distortions. Angular acceleration nonlinear distortions are of high frequency and therefore angular **rate** simulation errors do not usually exceed a pilot's sensitivity thresholds. Only specific forces caused by angular acceleration distortions are likely to exceed their thresholds. Thus false specific forces should form the basis for a fidelity criterion for angular motion. The permissible values of these false specific forces for small angular rates are similar to those shown in fig.9 for amplitude values less than 0.015g.

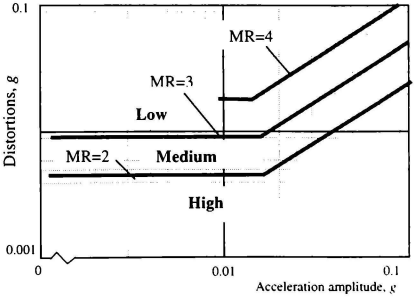


Fig.9 - The permissible values of nonlinear distortions.

Describing Function Distortions (DFD). Unlike nonlinear acceleration distortions, distortions in gain and phase can be perceived by a pilot in different ways. Firstly, certain values of these distortions can be perceived as a delay in the motion cues relative to the visual cues. Experiments show that the perception threshold value of this delay is about 0.03-0.05 sec at a frequency of 1Hz.

Secondly, DFDs can manifest themselves as a discrepancy in the characteristics of simulated and real aircraft. Investigations show that large distortions can affect piloting to a greater degree than switching the motion off (the upper curve in fig.10). They may even cause an Aircraft-Pilot Coupling even in cases where no oscillations arise on a fixed-base simulator.

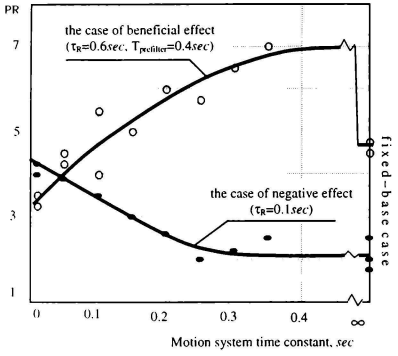


Fig.10 - DFD effect on motion fidelity.

The effect of the describing function depends on the role of accelerations in the case considered. Figure 10 shows that, where motion cues are beneficial, DFDs degrade the controllability of the simulated aircraft compared with the real one; for the negative acceleration effect, DFDs improve the controllability of the aircraft modeled. In the first case this effect is determined by phase lag, in the second by gain distortions.

The influence of DFD depends in a complicated way on aircraft characteristics and piloting task. It would be impossible to define this effect for all cases. It is reasonable to estimate permissible DFD values for those cases where the effect is greatest. Our results, for example those presented in fig.10, show that to avoid pilot ratings being affected, the motion system time delay should not exceed 0.03-0.05sec and gain distortions should not exceed 3dB for frequencies up to 3-5Hz.

Conclusions

Despite the fact that the technical characteristics of motion platform cueing devices have improved dramatically over the years, limited motion simulation fidelity remains one of the main factors which determines the potential of modern flight simulators and the validity of simulation results. Outstanding motion cueing problems are more of a scientific than a technical nature and are usually addressed by performing one-off experiments. A more widely applicable approach to estimating the effects of motion is required, including development of theoretical methods to estimate the effects of motion on piloting, optimisation of drive laws and identification of the requirements for motion cueing devices.

Recently TsAGI obtained some important results on these issues. A theoretical approach was created which adequately describes all existing experimental data on the effects of accelerations on lateral control. Criteria were developed which estimate the motion fidelity for different driving laws (washout, scaling, tilting). The requirements for nonlinear distortions and gain and phase distortions in reproducing specific forces and angular accelerations were also identified.

References

1. "Jane's Simulations and Training Systems", 1997-1998.
2. Advani S.K., Nahon M.A., Haeck N., Albronda J. "Optimization of Six-Degrees-of-Freedom Motion System for Flight Simulators", From AIAA Modeling and Simulation Technologies Conference, New Orleans, August 1997.
3. A.D.White, J.R.Hall, B.N.Tomlinson. "Initial Validation of an R&D Simulator With Large Amplitude Motion". AGARD-CP-513.
4. Schroeder J.A., "Evaluation of Simulation Motion Fidelity Criteria in the Vertical and Directional Axes", Journal of the American Helicopter Society", 1996.
5. J.R.Wood. "Comparison of Fixed-Base and In-Flight Simulation Results for Lateral High Order Systems". AIAA-83-2105, August 15-17, 1983, Gatlinburg, Tennessee.
6. McRuer D., "Human Dynamics in Man-Machine Systems'. Automatica., vol.16, No.3, 1980, pp.237-253.

7. V.V.Rodchenko, L.E.Zaichik, Yu.P.Yashin. "Motion Cueing on Research Flight Simulators". CEAS Symposium on Simulation Technology "Making it Real", Delft, The Netherlands, 30,31 Oct. - 1 Nov., 1995.
8. R.J.A.W.Hosman "Visual-Vestibular Interaction in the Perception and Control of Aircraft Motions by the Pilot". AIAA Modeling and Simulation Technologies Conference, New Orleans, August, 1997.
9. D.G.Mitchell, R.H.Hoh, J.M.Morgan, "Flight Investigation of Helicopter Low-Speed Response Requirements"' Journal of Guidance, Control, and Dynamics, vol.12, No.5, Sept.-Oct. 1989, pp.623-630.
10. Zaichik L.E., Rodchenko V.V., Rufov I.V., White A.D., Yashin Yu.P. «Acceleration Perception». From AIAA Flight Simulation Technologies Conference, Portland, August, 1999.
11. Sinacori J.B., "The Determination of Some Requirements for A Helicopter Flight Research Simulation Facility", NASA CR-152066, Sept 1977.
12. J.A.Schroeder, W.W.Y.Chung, D.T.Tran, S.Laforce, N.J.Bengford, "Pilot-Induced Oscillation Prediction with Three Levels of Simulation Motion Displacement", AIAA Modeling and Flight Simulation Technologies Conference, Boston, August, 1998.
13. Working Group: "Dynamic Characteristics for Flight Simulator Motion Systems", AGARD Advisory Report No.144, AGARD, NATO, Neuilly sur Seine, France, September 1979.
14. Military Standard MIL-STD-1558, Six-Degree-of – Freedom Motion System Requirements for Aircrewmember Training Simulators, February 1974.

EVALUATION OF A DYNAMIC SEAT AND A HELMET LOADER IN A MOTION-BASED FLIGHT SIMULATOR

Ir. Alex D.J. Rutten
National Aerospace Laboratory NLR, Amsterdam, The Netherlands

Abstract

The effects of a g-seat and a helmet loader on task performance and control behavior in longitudinal and lateral disturbance tasks was evaluated in the National Simulation Facility (NSF) of the National Aerospace Laboratories NLR. The goal of the research was to validate the g-cueing system using objective measures. In the end this research should result in optimizations of the drive laws of both g-cueing system and motion platform.

The effect of the g-cueing system in a longitudinal task was evaluated during two experiments in which 8 subjects participated. The results did not always show a clear effect of the g-cueing system and indicated that subjects responded differently to the application of g-cueing. It appears that the longitudinal cues from the seat may be ambivalent and improvement of the drive laws may be needed. Six subjects performed the lateral disturbance task. The effects of the g-seat are in line with the effects of the motion platform, but less strong. Additional measures should be taken to be able to assess the subjective realism of the g-cueing system.
For both the longitudinal and the lateral disturbance task, there were no additional significant effects of the combined use of motion and g-seat.

Introduction

The National Aerospace Laboratory NLR operates the National Simulation Facility (NSF) for research into handling qualities, training, flight control system design, flight simulation and human perception. The simulator is currently configured as an F-16 in MLU configuration, complete with a fully operable cockpit, mounted inside a 17 ft dome. The cockpit and dome are mounted on a 6 degrees of freedom motion platform.
The NLR acquired a g-cueing system from the French company Sogitec, for the National Simulation Facility (NSF) to complement the cues from the motion platform. The system comprises a g-seat (or dynamic seat), an active lap belt, a g-suit and a helmet loader and is installed in the F-16 MLU cockpit of the NSF. The system can be used in conjunction with the 6 degrees of freedom motion system, to provide a complete set of tactile and vestibular motion cues to the pilot.
Alternatively, the NSF can also be operated with the g-cueing system as the sole motion cueing device. Elements of the g-cueing system, like the seat, the helmet loader etc, can be de-selected, so the configuration of the g-cueing system can be tailored to suit experimental needs.
After installation of the g-cueing system in the simulator, the g-cueing system was used in a number of experiments that were not designed to evaluate motion cueing issues. Comments from pilots were positive, but often concentrated on the added realism of the cues of the g-suit. It should be remarked that the cues from the g-suit are very strong compared to the tactile cues from the seat and that cues from the seat are often masked by the cues from the suit.
Past research has shown distinct effects of motion cueing with dynamic seats. Control of roll and pitch was investigated for the ALCOGS seat[1]. Results showed that performance improved when seat cues were present. The seat was driven proportional to the zero and first derivatives of the roll and pitch angles.
Positive effects of a g-seat as a heave motion cueing device for helicopters were reported[2]. The seat was reported to enhance the realism of the simulation as well as improve the performance in certain tasks.
The design and control laws of the g-seat and helmet loader of the NLR are distinctly different from designs found in literature.
It was felt therefor, that in order to assess the actual effects of NLR's g-seat, specific experiments were needed that would look at the impact of the seat on task performance and control behavior. The experiments were designed to be a first validation of the principle and drive laws of the seat and helmet loader for pitch and roll control in a fighter simulation. Knowledge of the effects of the system can be used to optimize the control laws of the system and specifically, to determine the optimum way to combine the cues from the g-cueing system with the cues from the motion platform.

Copyright © 1999 by NLR. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

First of all, the experimental facility will be treated, emphasizing on the cueing systems. Subsequently, the longitudinal and lateral task experiments are described, followed by a discussion of the results and conclusions.

The National Simulation Facility

The National Simulation Facility (NSF) was used during the experiment. The NSF is currently configured for F-16 MLU simulations, with a 6 degrees of freedom motion platform, g-cueing devices, head tracked dome projection system and a fully operational F-16 MLU cockpit.

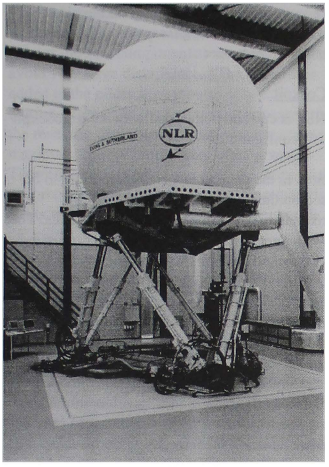


Figure 1 – The National Simulation Facility

The visual system

The NSF uses an Evans & Sutherland dome display system with a diameter of 17 ft. The image is projected by two lightvalve projectors, one for the low resolution background and one for the high resolution inset. The low resolution background has a field of view of approximately 140 degrees horizontally and 110 degrees vertically. The high resolution inset has a field of view of 50 by 35 degrees. The image is generated by an ESIG 3000 image generator. Various databases are available for this system. For the disturbance tasks, a very sharp horizon line was needed as a pitch and roll attitude indication of the aircraft. Therefor a generic ground/sky projection was used. This resulted in a very sharp horizon line and thus a very clear attitude indication.

The motion System

The motion system is a 6 degrees of freedom synergistic platform. The system bandwidth can be characterized by a phase lag of 45 degrees at 4 Hz.

The time delay between cues from the visual system and the motion system is in the order of 20 ms. The visual system lags behind the motion system.

A classical washout scheme is used for motion filtering with 2nd order high pass filters for roll and pitch accelerations and 3rd order high pass for specific force along the z-axis. The break frequencies of the motion rotational filters were at 0.67 Hz for all axes. The gains for all rotations were set at 0.2.
Specific Force Error Correction is used to compensate false cues due to rotational cueing. During the experiments, the end stops of the motion platform were not reached and the platform provided onset rotational cues for pitch, roll and yaw. Onset and sustained cues for specific forces in lateral and forward direction, as well as onset cues in vertical direction were generated by the platform as well.

The G-cueing system

The NLR G-cueing system consists of a g-seat with moving panels and inflatable cushions, an active lap belt, a helmet loader and a g-suit. For the experiment, the lap belt and the g-suit were not used.

Seat

The seat consists of a moving backrest, a moving seat panel and inflatable cushions in the backrest and seat panel. Below the cushions, metal bosses are located. When the pressure in the cushion decreases, the contact area with the metal bosses and the buttocks of the pilots increases, effectively lowering the seat comfort for the pilot. At 1g, the seat cushion is inflated to a level, just enough to keep the pilot of the metal bosses.

The cueing philosophy for the seat is a combination of position and pressure cueing. For example, under the influence of a normal g-load, like for instance in a turn or a pull-up maneuver, the pilot is pushed into his seat. The pilot experiences this through an increased pressure on the buttocks and through a lower eye point. In the simulator, this sensation is simulated by decreasing the pressure in the seat cushion, causing the pilot to get into contact with the metal parts of the seat, effectively 'hardening' the seat and by lowering the seat panel, thus lowering the pilot eye point.

The seat provides sustained and onset cues for pitch acceleration, forward and vertical specific force for longitudinal maneuvers. The vertical position of the seat panel is proportional to the pitch acceleration and the vertical specific force. The panel moves down and the seat cushion pressure is lowered for positive g-load.

For lateral maneuvers, the seat panel rotates and moves laterally. The sideways displacement of the seat is proportional to the lateral specific force. The rotation angle of the seat panel is directly proportional to the rotational acceleration and the rotational velocity in body axes of the aircraft. The seat thus provides both onset and sustained cues for rotations.

Helmet Loader

The helmet loader provides cues to vertical acceleration by pulling the helmet down under positive g-load. Wires are attached to a bar on the pilot helmet and feed through pulleys on the pilot harness to two torque motors mounted in the seat, behind the pilot. Because the bar is mounted on a pivot, the head movements of the pilot are not obstructed by the wires. Figure 2 shows the helmet with the helmet loader bar attached.

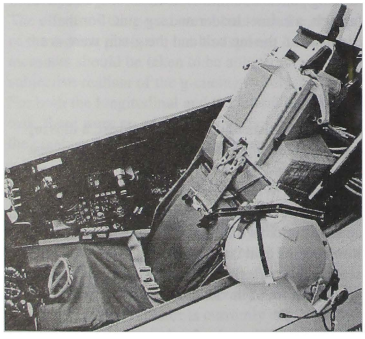


Figure 2 – G-seat and helmet with helmet loader bar installed

The helmet loader pressure is only a function of the normal acceleration at the pilot position. Table 1 summarizes the helmet loader drive law.

Acceleration [m/s^2]	Helmet load [N]
-10.00	0.00
0.00	0.00
9.81	9.0
30.00	27.0
85.00	45.0
100.0	45.0

Table 1 – Helmet loader drive law

Note that, at 1g, a constant pressure is applied to the helmet. This pressure keeps the wires tight and allows cueing at simulated g-levels between 0 and 1 g.
To prevent movement of the pilot harness in stead of movement of the helmet, care should be given to tighten the pilot harness.

The Cockpit

The NSF is currently equipped with a F-16 cockpit in MLU configuration.

Mounted in the cockpit is a HUD, modified to be focused on the dome surface, which is approximately 7.5 ft in front of the pilots eyes. The HUD can generate a reference symbol, consisting of two concentric circles with two perpendicular lines. This symbol was used as a reference. The symbol's position was adjusted by the subject before the start of each run, so the horizontal line coincided with the horizon projection of the visual system.

The F-16 force stick was used as an input device. During the disturbance tasks, inputs could only be given in one axis. The throttle, rudder pedals, trim wheels and trim switch were all disconnected for the purpose of the experiments.

Controlled Element Dynamics

One of the long-term goals of the experiments was to obtain an optimization of the motion cueing devices for the NSF. As the NSF is usually used for single seat fighter simulations, it was preferred to use the standard F-16 model in stead of a different dynamics for the controlled element.

Effects of G-Cueing during Longitudinal Disturbance tasks

Two separate trials were conducted to investigate the effects of the g-cueing system in longitudinal disturbance tasks. The first trial took place in November 1997 and compared the effects of the g-seat and helmet loader, with fixed based and motion based configurations. The follow-on trial took place

in November 1998 and was designed to verify the findings of the previous trial and to look at the effect of the seat without the helmet loader.

During both trials, a longitudinal disturbance task was used. The subjects had to maintain a steady pitch attitude under the influence of disturbances. Before the start of a run, subjects had to adjust the reference symbol in the HUD and align it with the horizon as projected by the visual system. This would be the reference for the duration of the run. The subjects had to keep the visual horizon aligned with the reference symbol under the influence of disturbances.

The disturbance signal consisted of a sum of 10 sines in the frequency range of 0.049 to 2.5 Hz. The gain of the sine components was either 1 or –1, to avoid very large disturbances. The signal was added to the pilots stick inputs.
During analyses of the first trial, it appeared that the resulting disturbance signal was too strong, as maximum control inputs were often given. Therefor, during the second trial, the gains of the disturbance signal were reduced. Consequently, no limits were reached in the control inputs made by the pilots.

All data was sampled with 50 Hz. The duration of the run was limited to 50 seconds, of which 2048 samples were used for the data analyses.

Configurations

The motion condition of the simulator was the controlled variable in the experiments. In total, 5 different motion conditions of the simulator were evaluated and are summarized in the table below.

	Configuration				
	1	2	3	4	5
G-seat	**On**	Off	Off	**On**	**On**
Motion	**On**	**On**	Off	Off	Off
Helmet loader	**On**	Off	Off	**On**	Off
	1	1	1, 2	1, 2	2
	Trial				

Table 2 - Configurations evaluated during two trials

Configurations 2 and 3 (platform motion and fixed base) were included to act as baselines, against which the results of the g-seat and helmet loader could be compared.
The second trial was designed to evaluate the g-seat separately from the helmet loader and to be a validation of the first trial. Although configurations 3 and 4 were evaluated in both trials, a direct comparison between the second and first trial was not possible. Reasons for this are the appearance of limits in the pilot control inputs during the first trial and a disturbance signal with reduced gain during the second trial. Another reason is a different initial condition of the F-16 model for the second trial and a different balancing of configurations over the subjects.

Experimental Design

During the first trial, 4 subjects performed 11 runs in 4 configurations, so a total of 176 runs were performed.
During the second trial, 4 subjects performed 12 runs in 3 configurations, so a total of 144 runs were performed.

Every run lasted approximately 50 seconds and was terminated by the experiment controller. The disturbance signal remained constant after all sine components had completed an integer number of cycles.

The configurations were balanced over the subjects. During the first trial every new run presented a change in configuration. During the second trial, the seat was to be used separately from the helmet loader. Because engaging and disengaging the helmet loader took some time, a frequent change in configuration was considered cumbersome. Therefor, the configurations were grouped in blocks of 4 runs, during the second trial, so that changes in configuration took place after 4 runs. Runs were consequently balanced over the subjects in blocks of 4 runs.

Although not considered essential, all participants had previous flying experience in light aircraft. Breaks were given every 20 minutes, or more often if a subject was getting tired.

Analysis of the data

Effects on task performance and control behavior were considered.
For both trials, the results of the subjects were combined and an Anova was used to determine the significance of the effects. For the first trial a level of significance of 0.09 was used and the effects of the variables g-cueing system, motion system and training were determined as well as the combined effects of those variables.
For the second trial, a more elaborate analysis was performed with a level of significance of 0.05. The significance of effects from configuration, training, subject and various interactions were analyzed.

Task performance

The task the pilots had to perform consisted of keeping the reference point in the HUD on the visual horizon. The task was thus essentially to keep the pitch angle deviations as small as possible. Standard deviations of theta and pitch velocity and pitch acceleration were analyzed.

After averaging the results over all four subjects a significant decrease of the standard deviation of the pitch attitude could be observed when either the motion system was switched on, or the g-cueing system was switched on. Combined operation of the motion platform and the g-seat/helmet loader did not yield a significant change in the standard deviation of the pitch angle although the trend shown in figure 3 suggests differently. The vertical bars in figure 3 indicate plus and minus one standard deviation. The results shown are from the first trial.

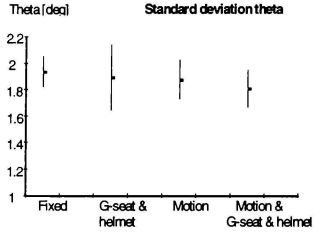


Figure 3 - Effect of configuration on the standard deviation of the pitch angle

During the second trial, the same trend, although not significant, could be observed; a decrease in the standard deviation of the pitch angle when a form of cueing was added. During the second trial, also a combined effect of subject and configuration was observed, which could indicate that subjects reacted differently to the application of the cueing device.

The body pitch velocity exhibited a significant increase when either motion or g-cueing was added (figure 4). During the second trial, the results were not that clear and the application of the g-seat did not result in a significant change in the standard deviation of the pitch velocity. Also, a significant interaction between the motion configuration and subjects was found. This again indicates that subjects reacted differently to the cues from the g-cueing system.

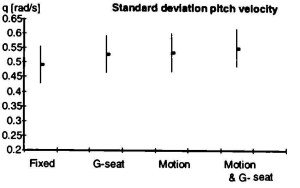


Figure 4 - Effect of configuration on the standard deviation of the pitch velocity

During the first trial, a significant change in the standard deviation of the body pitch acceleration due to the motion configuration was measured; changes in configuration between fixed base, g-cueing, motion and motion with g-cueing yielded different averaged body pitch accelerations (figure 5).

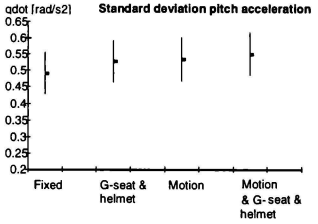


Figure 5 - Effect of cueing configuration on the standard deviation of the pitch acceleration

This effect was also confirmed during the second trial, as can be seen in figure 6.

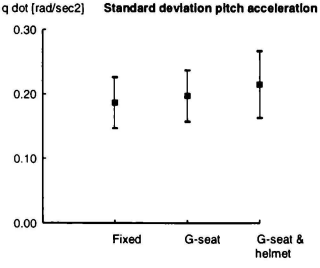


Figure 6 - Effect of cueing configuration on pitch acceleration during the second trial

The decrease in the standard deviation of the specific force Az due to the application of the g-seat/helmet loader proved to be significant during the first trial (figure 7). During the second trial, this effect was confirmed (figure 8). The interaction of configuration with subjects proved to be significant during the second trial. This implies that the subjects responded differently to the application of the g-seat and helmet loader.

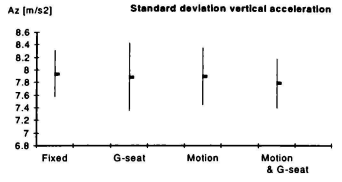


Figure 7 – Effect of configuration on the standard deviation of the vertical acceleration (1st trial)

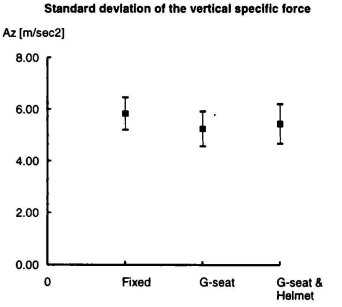


Figure 8 – Influence of configuration the standard deviation of the vertical acceleration (2nd trial)

Control Behavior

Analysis of the control behavior of the first trial took place in the frequency domain and the time domain. As a measure in the time domain, the standard deviation of the longitudinal stick inputs was used (figure 9).

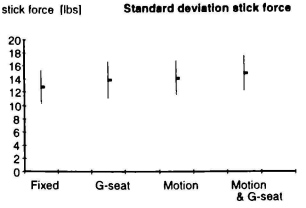


Figure 9 – Effect of cueing configuration on the standard deviation of the longitudinal stick force

When the results were averaged over all subjects, the standard deviation of the longitudinal stick inputs increased significantly when motion or g-cueing was applied. Combined use of motion and g-cueing did not cause a significant improvement over a situation with motion or g-cueing alone. During the second trial, the application of g-cueing seemed to increase the standard deviation of the control force, although this effect was not significant (figure 10).

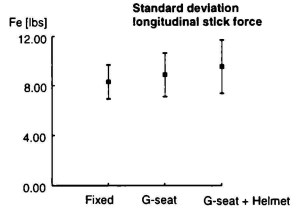


Figure 10 - Effect of the cueing configuration on stick inputs

In the frequency domain analysis of the first trial, the cut-off frequency of the longitudinal stick inputs was used. The cut-off frequency is defined as the frequency below which half of the power of the signal is contained[3]. The results in the frequency domain seemed to indicate an increase in the cut-off frequency when motion and g-cueing were applied. However, the results are disturbed by the fact that during most runs the stops in the longitudinal stick input were reached.

Effects of G-Cueing during Lateral Disturbance Tasks

In 1998 a trial was conducted to evaluate the effects of cues from the motion platform and the g-seat in a lateral disturbance task. The subjects had to null any deviations from the wings level roll attitude under the influence of disturbances. To judge the roll attitude, a static reference symbol in the HUD was provided.

The disturbance signal consisted of a sum of 10 sines in the frequency range of 0.049 to 2.5 Hz. The signal was added to the pilots stick inputs. A complete number of cycles of every sine component was completed during the total run.

All data was sampled with 50 Hz. The duration of the run was limited to 50 seconds, of which 2048 samples were used for the data analyses.

Configurations

Again, the motion condition of the simulator was the controlled variable in the experiment. Four combinations of motion and g-cueing were evaluated. The seat was used without the helmet loader in the trial, because the seat is the only component of the g-cueing system that provides any cues for lateral maneuvers. Table 3 summarizes the configurations used in the experiment.

	Configuration			
	Fixed	*G-seat*	*Motion*	*G-seat & motion*
G-seat	Off	**On**	Off	**On**
Motion	Off	Off	**On**	**On**

Table 3 - Motion conditions evaluated during the lateral disturbance task

The experiment was designed to be a preliminary evaluation of the effects of a g-seat in a lateral disturbance task. Configurations with fixed base and with motion platform, were included to act as a baseline for the configuration with the g-seat.

Experimental Design

Six subjects participated in the trial. Each subject performed 6 runs in all 4 configurations. A total of 144 runs were made.
The configurations were balanced over the subjects. After every run, the configuration (the motion condition) of the simulator was changed.

Analysis of the data

The analysis of the data was done in the time domain. The results of the subjects were combined and an Anova was used to determine the significance of the effects. A level of significance of 0.05 was used and the effects of the variables g-seat, subject and motion system were determined as well as the combined effects of those variables.

Task Performance

The task consisted of keeping the deviations of the roll angle as small as possible around a wings level attitude. The task performance was therefor primarily evaluated by looking at the standard deviations of the roll angle. The influence of g-cueing and motion cues on roll velocity and roll acceleration was also considered.

The presence of cues from the g-seat caused a significant reduction in the standard deviation of the roll angle when compared to a fixed base configuration. Also the presence of motion cues reduced the standard deviation of the roll angle (figure 11). There was no significant effect from the combined use of motion and g-cueing nor were there any effects from the interaction between subject and configuration.

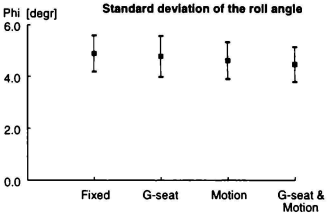


Figure 11 - Influence of cueing configuration on the standard deviation of the roll angle

The roll velocity was not significantly influenced by the experimental conditions (figure 12).

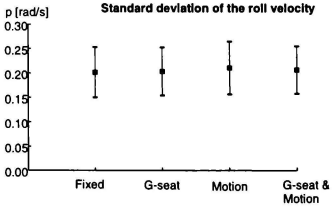


Figure 12 - Influence of cueing configuration on the standard deviation of the roll velocity

The standard deviation of the roll acceleration increased however, when the motion platform was

active. No significant change resulted from the application of g-cueing (figure 13).

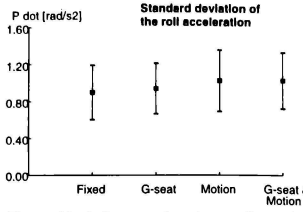


Figure 13 - Influence of cueing configuration on the standard deviation of the roll acceleration

Control Behavior

The control behavior was analyzed by evaluating the influence of motion and g-seat on the standard deviation of the lateral control forces. The standard deviation of the lateral control force increased significantly when cues from the motion platform were added (figure 14). No comparable influence from cues from the g-cueing system was present. There was a significant interaction between subjects, motion and g-cueing. This implies that subjects responded differently to the application of combined motion and g-cueing.

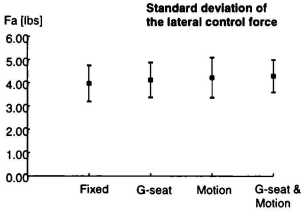


Figure 14 - Influence of cueing configuration on the stick force

Discussion

The application of the g-cueing system resulted in significant changes in task performance and control behavior in the lateral and longitudinal disturbance tasks.

For most parameters, except for the standard deviation of the body pitch acceleration, no statistically significant effect of the *combined use* of g-cueing and platform motion could be found. This may be due to the fact that both systems give the same type of feedback and that consequently, there is no benefit to be obtained in providing the same cue with both systems. On the other hand, the drive laws of the seat were not optimized and different results may be obtained when an optimization of the drive laws is performed.

Although the first experiment with the longitudinal disturbance task showed significant changes in task performance and control behavior, those results could not be completely confirmed during the second experiment. This may be partly due to the different experiment set-up.

In addition, in the second experiment, there was an interaction between subjects and configuration, indicating that subjects responded differently to the application of g-cueing.

Some of those findings may be explained by the difference in subjects, but the conflict between the sustained cue and the onset cue of the seat might be another cause. As explained earlier, the seat moves down at positive g, such as for example in a pitch up maneuver. In an aircraft, or in a simulator with a motion platform, a pilot would sense an upward acceleration. In the g-seat, the pilot is moved down in a pitch up maneuver. In the end, this provides the right cue (a lower eye point), but initially the acceleration is in the wrong direction. Optimizing the seat for onset cueing and sustained cueing in this direction will need some more experimentation. One solution could be to use the cushions for sustained cueing and to use the seat panel for onset cueing. The seat panel should be driven in the same direction as the motion platform, i.e. up for positive Gz load. The upward motion would consequently have to be washed out. This would imply that the seat panel would be driven like a 'miniature motion platform'.

In the lateral task, cues from the motion system resulted in changes in the standard deviation of the roll angle, the roll acceleration and the stick force. Although trends indicate an influence comparable to that of the motion system, the application of cues from the g-cueing system had a less strong effect and only influenced the roll angle significantly. When simulating lateral specific forces, the seat panel moves sideways and rotates. When entering a left turn for example, the seat panel is translated to the left and rotated to the left. The pilot will feel translated to the left and he will sense an increased pressure on his right buttock. These cues are in the same direction as the cues he receives in the aircraft.

Although the response of the pilot to the lateral cue is comparable to his response to a cue from the motion system, it is not clear whether the cue is 'realistic' and if training with the seat would transfer to the aircraft. The experiments conducted

only evaluated the measurable effects on control behavior and task performance. The realism of the cues from the g-seat or the possible transfer of training that can be obtained with the device, were not evaluated.

Conclusions

The application of the g-cueing system resulted in significant changes in task performance and control behavior in lateral and longitudinal disturbance tasks. For both the longitudinal and the lateral disturbance tasks, there were no significant effects of the combined use of motion and g-seat.
In the longitudinal disturbance task, there is reason to believe that cues from the seat are ambivalent. Modifications will have to be done to the drive laws of the seat in order to reduce this cue conflict and to obtain a better integration with the cues from the motion platform.
For the lateral disturbance task, the effects of the g-seat were more in line with the effects of the motion platform. The lateral cues of the seat appear to be correct. As a next step, the subjective realism of the cues of the g-seat would have to be assessed, to obtain a complete view of the functioning of the seat.

References

1. Cress, J.D., MacMillan, G.R., Gilkey, M.J. (1989). The dynamic seat as an angular cuing device: control of roll and pitch vs. the control of altitude and heading. Presented at AIAA Flight Simulation Technologies Conference, Boston, MA.
2. White, A.D. (1989). G-seat heave motion cueing for improved handling in helicopter simulators. Presented at AIAA Flight Simulation Technologies Conference, Boston, MA.
3. White, A.D. (1994). Measures of flight simulator fidelity based on pilot control behaviour. Presented at the International Training Equipment Conference, The Hague, The Netherlands.

AIAA-99-4333

PRELIMINARY INVESTIGATION OF THE MOTION FIDELITY CRITERION FOR A PITCH-LONGITUDINAL TRANSLATIONAL TASK

Duc T. Tran*
NASA Ames Research Center
Moffett Field, California

William W. Chung†
Logicon Information System
Moffett Field, California

Julie Mikula†
NASA Ames Research Center
Moffett Field, California

Abstract

While the ground-based flight simulation community is increasingly interested in determining the required motion fidelity to meet mission requirements, little has been done to determine the motion fidelity requirements for the pitch and longitudinal degrees-of-freedom. Typically, the pitch and longitudinal motion fidelity is determined by following the same criteria as the roll and lateral degrees-of-freedom. However, pitch visual cues are typically different from roll visual cues. Since visual and motion cues are known to interact, investigation of the pitch and longitudinal motion fidelity criteria is warranted

This investigation used a helicopter model with satisfactory handling qualities to translate longitudinally between two points 20 feet apart. A simulation cueing baseline was developed such that the motion and visual cues were one to one. Motion cueing reductions were subsequently evaluated from this baseline case. Two angular motion and eleven translational motion configurations were flown by four test pilots. Results indicate that as longitudinal phase distortion is increased, motion fidelity ratings decreases and also as longitudinal motion gain decreases, the motion fidelity ratings also decrease. Both results are consistent with the roll-lateral motion requirements.

Nomenclature

g	gravitational acceleration, ft / sec^2
$M_{\delta_{lon}}$	pitch control power, rad / sec^2 / in
M_q	pitch acceleration due to pitch rate, 1 / sec
q	pitch angular rate, rad / sec
$\dot{q}$	pitch angular acceleration, rad / sec^2
r_z	vertical displacement between pilot' s abdomen and simulator rotational center, positive down, ft
u	groundspeed along x - body axis, ft / sec
$\dot{u}$	vehicle acceleration along x - body axis, ft / sec^2
δ_{lon}	pilot longitudinal stick input, in.
θ	pitch attitude, rad
$\dot{\theta}$	pitch rate, rad / sec

* Senior Member of AIAA.

† Member of AIAA.

Copyright © 1999 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

Introduction

Studies have been conducted to investigate the cueing fidelity requirements for roll-lateral interactions but little has been done for the pitch-longitudinal interactions[1-3]. From a psychophysics point of view, there is little perceptual difference as provided by a human's angular sensors[4]. In Zacharias's[5] assessment of the pilot's motion sensor models, the mathematical treatment of the roll and pitch angular rate sensing characteristics are also similar. However, in most motion-based simulators, the pitch and longitudinal visual cues are typically poor as compared to roll-lateral visual cues. Vertical field-of-view (FOV) is usually more constrained so that the horizon can disappear taking away the pilot's pitch attitude awareness. Also little ground is seen in front or below the pilot providing little if any longitudinal position awareness. Depth cues are also typically poor, thus damaging the pilot's longitudinal awareness. Furthermore, from a tactile consideration, the pilot's contacts with the seat and restraint system in the pitch-longitudinal axes are different from the roll-lateral axes. Taking these differences into account, the cueing fidelity requirements for pitch-longitudinal interactions need to be examined in their own right. This investigation was undertaken to provide some preliminary insights into the motion requirements for combined pitch-longitudinal motion.

In flight simulation, pitch and longitudinal motion

need to be treated together. For a coordinated maneuver, longitudinal platform motion provides an acceleration to counteract the "leans" that would result from only pitching the cockpit. Typically, the longitudinal motion used for this purpose does not eliminate the leans completely but is conventionally attenuated through a washout filter to maintain the simulator within its physical travel limit. This compromise introduces an erroneous specific force[6] depending on the magnitude of the pitch attitude. This investigation examined the effects of this erroneous cue and compared the results to the roll-lateral requirements.

Experiment Description

Aircraft Model, Force Characteristics, and The Task

A two DOF helicopter model with a pitch rate command and a fully coordinated pitch-longitudinal response about hover is given by equation 1,

$$\begin{bmatrix} \dot{q} \\ \dot{u} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} M_q & 0 & 0 \\ 0 & 0 & -g \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ u \\ \theta \end{bmatrix} + \begin{bmatrix} M_{\delta_{lon}} \\ 0 \\ 0 \end{bmatrix} \delta_{lon} \qquad (1)$$

The pitch acceleration due to pitch rate (or the pitch damping stability derivative), M_q, was set at –4.0 1/sec and the pitch control power, $M_{\delta lon}$, was 0.67 rad/sec^2/in. The rotational center was set at the pilot's abdomen. The pitch rate response to the stick was developed to have the satisfactory handling qualities according to the Reference 7 specification. The longitudinal stick force feel characteristics was set at 2 lb/in with a 0.5 lb breakout force.

The Task

The task was a 20 ft dash-and-stop task performed at a constant altitude of 23 ft as shown in Figure 1 along the center of a runway. The piloted task started with a 20 ft translation towards the desired hover position situated in front of the helicopter's initial hover position, followed by 20 seconds of stationkeeping. The stationkeeping point was denoted by a series of four walls located 100 ft to the right and 100 ft in front and positioned at an angle of 45 degrees. At the desired stationkeeping point, the pilot should see the walls "edge on" out the right window. At the proper position, a building in the background appeared in the middle of the two inner walls. The walls were spaced such that the distance between the two inner walls translated to a distance of ±5 ft with respect to the desired stationkeeping point in the axis of travel. Likewise, the distance between the two outer walls translated to a distance of ± 10 ft with respect to the desired stationkeeping point in the axis of travel. A bob-up target was placed in the center of the runway and various red cones were placed on the runway visible through the right chin window for additional visual cueing.

Pilots were instructed to complete the dash-and-stop task in one smooth maneuver within the performance standards shown in Table 1. A sum-of-sines, Table 2, was added to the pilot's stick input to simulate wind gusts.

Procedure

Four experienced test pilots participated in the test. Each pilot practiced with randomly selected motion configurations at the beginning of each session. During the data collection, each pilot evaluated the motion configurations in a random order. The pilots were asked to fly each motion configuration three times and to evaluate the third repetition. Pilots were asked to give handling qualities ratings (HQRs)[8] and motion fidelity scale (MFS) ratings as shown in Table 3. A questionnaire was also given to elicit comments from the pilots regarding control strategy and motion fidelity.

Test Facilities

The NASA Ames Vertical Motion Simulator (VMS) was used for this investigation, which is shown in Figure 2. The cockpit was oriented such that the longitudinal axis of the simulator cab traveled along the beam which has a usable travel of 40 ft. The large longitudinal travel of the VMS allowed for the 20 ft dash-and-stop task without any motion cue attenuation. The motion system responses were collected using white noise with a Gaussian distribution and checked with frequency response technique developed for system identification called CIFER®[9]. For this investigation, each motion axis had an equivalent time delay of 60 msec. The visual FOV of the helicopter cab is shown in Figure 4.

The visual system was an Evans and Sutherland ESIG 4530 image generator. The out-the-window scene was presented on four monitors with a collimation/beam-splitter system. The visual system time delay was measured at 60 msec, which matches the equivalent time delay in the pitch and longitudinal motion axes.

Test Configurations

Two second-order washout filters, Figure 3, were developed for this investigation. A pitch washout filter generated the pitch motion commands, and a coordinated longitudinal washout filter provided the longitudinal motion to counteract the leans due to pitch. Two pitch washout filter configurations and eleven longitudinal washout filter configurations, shown in Tables 3 and 4 respectively and presented in Figure 5, were selected to investigate the interaction between pitch motion and longitudinal motion. Each configuration consisted of the gain and filter frequency for the respective washout filter. The configurations were

selected to represent low, medium, and high motion fidelity as defined in Reference 10.

Results

Four pilots took part in this test. Their average MFS ratings and the average HQR are shown in Figures 6 and 7 respectively. All pilots rated the fixed-base case as low fidelity. In general, the amount of longitudinal platform motion increases as both the longitudinal filter's predicted fidelity and the pitch filter's predicted fidelity increases, i.e., increasing motion gain and decreasing phase distortion. However, as the amount of required longitudinal motion increases any parasitic differences in the dynamics between the pitch and longitudinal motion become exacerbated and potentially objectionable. Pilots commented on the sharpness of the motion cues due to the high gains. This is evident in the full-motion case (A1, T1) in that the pilots felt the motion cues to be objectionable and rated it as low fidelity. As pitch motion is attenuated (A3, T1), the longitudinal platform motion decreased and artifacts such as above were reduced significantly which resulted in an improved motion fidelity rating.

In both the full and the attenuated pitch motion cases, the motion cues became objectionable when the longitudinal phase distortion was 80 degrees at 1 rad/sec ($\omega_x = 0.9$). Figure 8 shows the average MFS as a function of ω_x. As expected, the MFS decreases as longitudinal phase distortion is increased regardless of pitch motion. This is consistent with the roll-lateral motion requirements.

In both the full and the attenuated pitch motion cases, it is clear that pilots felt the motion cues to be objectionable when little coordinated motion was provided, i.e., cases where the longitudinal motion gain was at or below 0.2. This is consistent with Schroeder[2] and Chung[3] in their roll-lateral motion fidelity requirement investigations. Figure 9 shows the average MFS as a function of G_x. For the attenuated pitch motion case (A3), MFS ratings generally decrease as longitudinal motion gain decreases. Again this is consistent with roll-lateral motion requirements. For the full pitch motion case (A1), there is no clear trend due to the low rating for the (A1, T3) case where it is expected to be rated medium fidelity. Undesired motion artifacts might have played a role in receiving the low fidelity rating while the longitudinal motion gain is still considerably high at 0.8.

One observation is made from Figure 6 when comparing the average MFS between the full pitch motion (A1) and the attenuated pitch motion (A3). For the cases that represent the medium and high fidelity region, i.e., cases T2, T3, T5, T6, the average MFS ratings for the attenuated pitch motion case are noticeably improved when compared with the full pitch motion case. This improvement may be contributed to less pitch motion resulting in less "leans". Therefore any objectionable situations would be expected to occur less frequently, and consequently result in the noticeable improvement in MFS ratings.

For both the full and attenuated pitch motion cases, the averaged HQR consistently followed the averaged motion fidelity ratings, i.e., HQR improves when MFS improves.

Concluding Remarks

1) Large phase distortion in longitudinal motion due to the pitch motion is detrimental to the motion fidelity. Pilots found it to be objectionable. This is consistent with the roll-lateral motion requirements.

2) Ratings generally improved as the longitudinal motion gain increased until the undesired motion artifacts became noticeable.

3) Additional testing is recommended to further investigate the pitch-longitudinal motion cueing fidelity requirements.

References

[1]Jex, H.R., Jewell, W.F., Magdaleno, R.E., and Junker, A.M.: "Effects of Various Lateral-Beam Washouts on Pilot Tracking and Opionio in the Lamar Simulator," AFFDL-TR-79-3134, pp. 244-266.

[2]Schroeder, J.A. and Chung, W.Y.: "Effects of Roll and Lateral Flight Simulation Motion Gains on a Sidestep Task," American Helicopter Society's 53rd Annual Forum, April 1997.

[3]Chung, W.Y., Robinson, D.J., Wong, J., and Tran, D.: "Investigation of Roll-Lateral Coordinated Motion Requirements with a Conventional Hexapod Motion Platform," AIAA Motion and Simulation Techonologies Conference, AIAA-98-4172, August 1998.

[4]A.J. Benson, et al: "Vestibular System Part 2: Psychophysics, Applied Aspects and General Interpretations," Springer-Verlag New York 1974, pp. 501-502.

[5]Zacharias, G.L.: "Motion Cue Models for Pilot Vehicle Analysis," AMRL TR-78-2, 1978.

[6]Grant, P.R. and Reid, L.D.: "Motion Washout Filter Tuning: Rules and Requirements," AIAA Flight Simulation Technologies Conference, August 1995.

[7]Aeronautical Design Standard, Handling Qualities

Requirements for Military Rotorcraft, ADS-33D, St. Louis, MO, July 1994.

[8]Cooper, G. E., and Harper, R. P., Jr.: "The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities," NASA TN D-5153, April 1969.

[9]Tischler, M. B., Cauffman, M.G.: "Frequency-Response Method for Rotorcraft System Identification: Flight Applications to BO-105 Coupled Rotor/Fuselage Dynamics," *Journal of the American Helicopter Society*, Vol. 37, No. 3, pgs. 3-17, July 1992.

[10]Sinacori, J. B.: "The Determination of Some Requirements for a Helicopter Flight Research Simulation Facility," NASA CR 152066, September 1977.

Table 1. Task performance standards

	Desired	Adequate
Longitudinal translation completion time	7 sec	11 sec
Stationkeeping position tolerance	+/- 5 ft	+/- 10 ft

Table 2. External disturbance

Frequency (rad/sec)	0.28	0.49	0.80	1.50	2.67	4.63	8.50
Amplitude (in)	0.002	0.006	0.014	0.032	0.054	0.068	0.06

Table 3. Motion fidelity scale

	Description	Score
High Fidelity	Motion sensations are not noticeably different from those of visual flight	3
Medium Fidelity	Motion sensations are noticeably different from those of visual flight, but not objectionable	2
Low Fidelity	Motion sensations are noticeably different from those of visual flight and objectionable	1

Table 4. Angular motion washout gains and filter frequencies configurations

Pitch Motion Configurations	Motion Gain G_q	Washout Filter Frequency, ω_q (rad/sec)	@ 1 rad/sec	
			Gain	Phase distortion (degree)
A1	1.0	0.0001	1	0
A3	0.6	0.5	0.58	43

Table 5. Translational washout gains and filter frequencies configurations

Longitudinal Motion Configurations	Motion Gain G_x	Washout Filter Frequency, ω_x (rad/sec)	@ 1 rad/sec	
			Gain	Phase distortion (degree)
T1	1.0	0.0001	1.0	0
T2	0.8	0.25	0.8	20
T3	0.8	0.5	0.78	43
T4	0.8	0.9	0.63	80
T5	0.5	0.25	0.5	20
T6	0.5	0.5	0.49	43
T7	0.5	0.9	0.4	80
T8	0.2	0.25	0.2	20
T9	0.2	0.5	0.2	43
T10	0.2	0.9	0.16	80
T11 (fixed base)	0.0	0.0	0.0	0

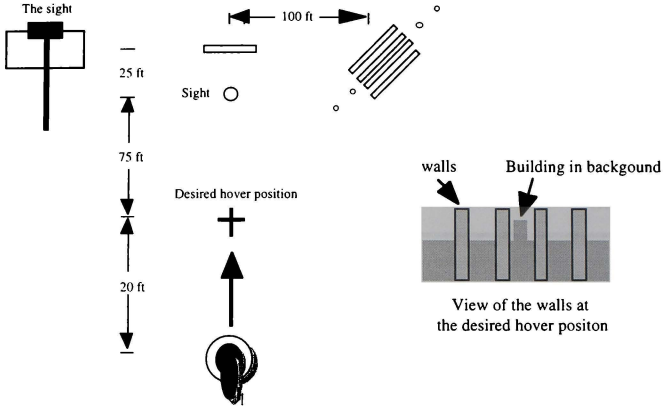
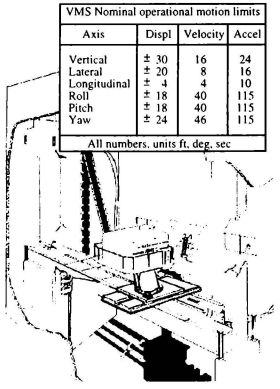


Figure 1. The longitudinal dash-and-stop task



VMS Nominal operational motion limits			
Axis	Displ	Velocity	Accel
Vertical	± 30	16	24
Lateral	± 20	8	16
Longitudinal	± 4	4	10
Roll	± 18	40	115
Pitch	± 18	40	115
Yaw	± 24	46	115
All numbers, units ft, deg, sec			

Figure 2. Vertical Motion Simulator (VMS)

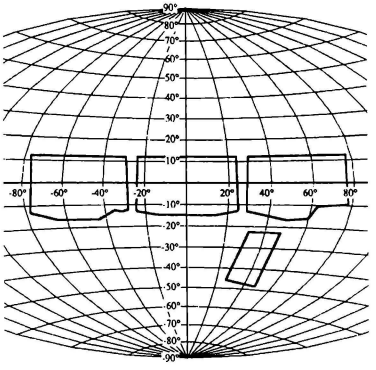


Figure 3. VMS cockpit field-of-view

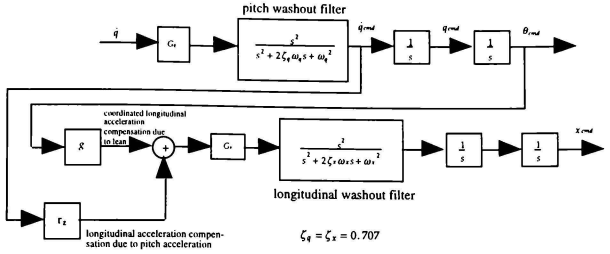


Figure 4. A representative motion drive command block diagram for pitch and longitudinal drives.

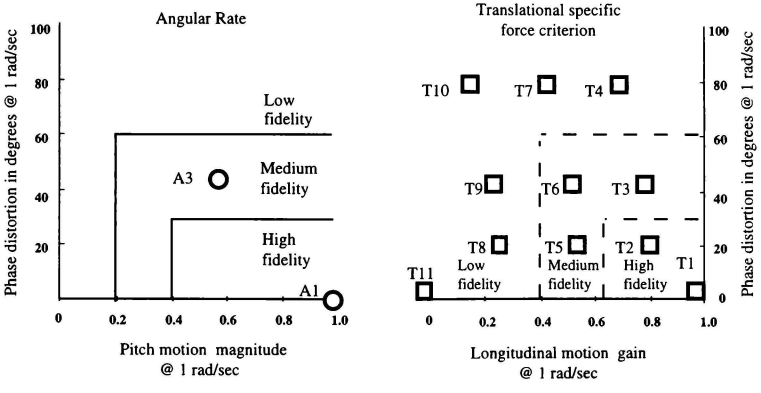


Figure 5. Test motion washout filter configurations for the pitch and coordinated longitudinal motion

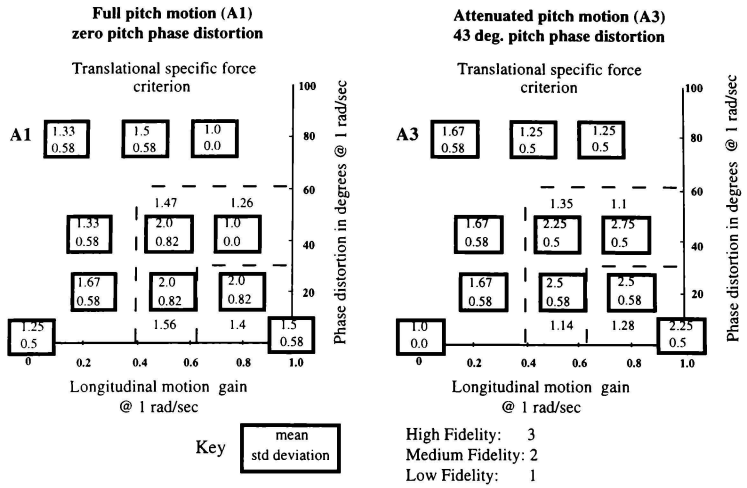


Figure 6. The average MFS

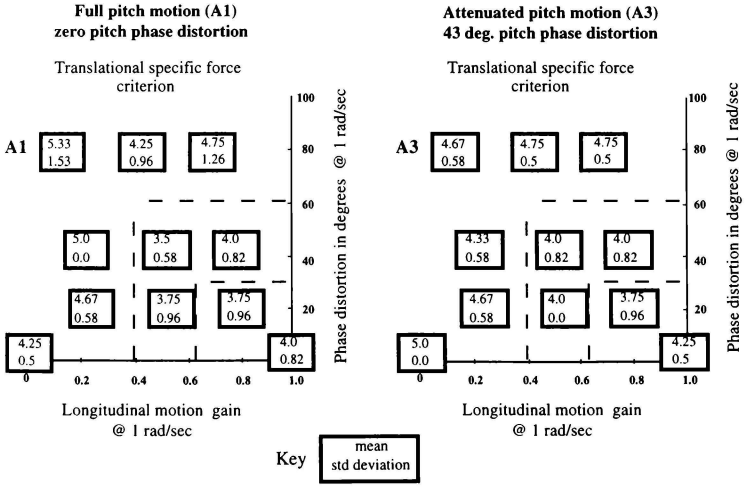


Figure 7. The average HQR

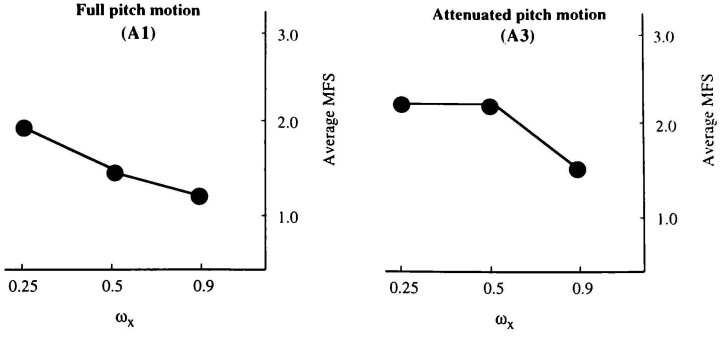


Figure 8. The average MFS vs. ω_x

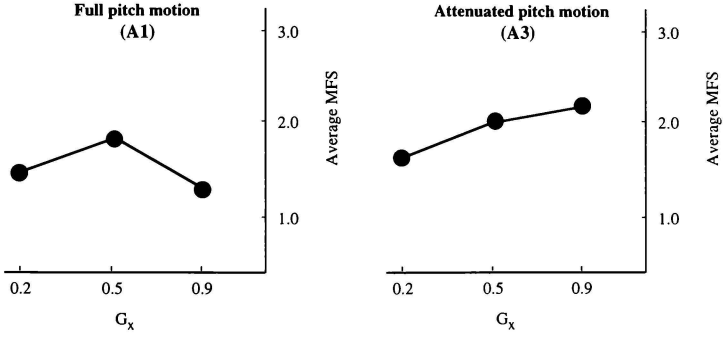


Figure 9. The average MFS vs. G_x

ACCELERATION PERCEPTION

L.E.Zaichik[1], V.V.Rodchenko[2], I.V.Rufov[3], Y.P.Yashin[4]
Handling Qualities and Flight Simulation Section
Central Aerohydrodynamic Institute (TsAGI)
Zhukovsky, Russia

A.D.White[5]
Defence Evaluation and Research Agency (DERA)
Bedford, United Kingdom

Abstract

The results of simulator experiments on acceleration perception using a 6-dof motion platform are presented and compared with previously published data. The roles of different sensory systems (vestibular, kinesthetic, tactile) in perception of the accelerations are analysed and acceleration perception math-models are discussed. Absolute and differential sensitivity thresholds are determined for sinusoidal accelerations of different frequencies and the influence of different factors on the threshold values is estimated. The laws governing the perception of acceleration over-threshold values are substantiated.

Introduction

Knowledge of the motion perception process is important for the solution of various motion cueing problems. This knowledge is used, for example, in Ref.1 to create a theoretical approach to estimating the acceleration roles in piloting, to develop criteria of motion system driving laws, and to identify the requirements for motion platform dynamics. Despite the fact that great attention has been paid to studying the process of pilot acceleration perception, this process is still not sufficiently well understood.

Difficulties in studying the acceleration perception mechanisms can be attributed to their complexity, since the central neural system (CNS), vestibular organs and the multitude of tactile and kinesthetic receptors (so-called mechanoreceptors) participate in this process. Mechanoreceptors are very complex, multiple and scattered all over the body. At present only vestibular organs and some mechanoreceptors have been studied by measuring physical characteristics and bioelectrical signals. Thus acceleration perception is difficult to investigate using objective methods. The only parameter which characterises the perception process as a whole is a subject's opinion of his own sensations. Such data can be obtained only by experiment.

The acceleration perception system can be assumed to be a sensor whose output signal is used by a pilot to control an aircraft. To analyse any control system we need to know the static and dynamic characteristics of the sensor and its dead zone. Similarly, to solve motion cueing problems we need to understand the following issues: (1) what math-model can adequately describe the dynamics of the acceleration perception system? (2) what are the human

[1] Leading Research Engineer
[2] Head of Section
[3] Engineer
[4] Leading Research Engineer
[5] Principal Scientist, Flight Management and Control Department

© British Crown copyright 1999. Published with the permission of DERA on behalf of the Controller of HMSO. Copyright © 1999 by TsAGI. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

acceleration sensitivity thresholds? (3) what is the relationship between accelerations and the intensity of sensations?

Existing dynamic math-models are based mainly on vestibular system analysis, since the vestibular organs' structure has been well studied[2-7]. However, these models are not sufficient to adequately describe the acceleration perception process, since other human sensory systems also participate in the process. Therefore, to further improve acceleration perception math-models we need to know the roles of the various sensory systems in acceleration perception.

The most comprehensive data on acceleration sensitivity thresholds were obtained by R.Hosman[8]. However, the data needs supplying and specifying, since the thresholds were obtained for only three degrees of freedom and without compensation for the false specific forces arising while cockpit tilting. Publications available on the subject (Ref.8 included) usually demonstrate threshold values for a single degree of freedom but, in real flight, specific forces and angular accelerations act simultaneously. For example, for a conventional fixed-wing aircraft configuration, pitch motion and g-load occur simultaneously as the manipulator is deflected. Thus, we need to understand how these specific forces influence pilot sensitivity to angular motion.

In simulators, increasing levels of reproduced acceleration are accompanied by increasing levels of acceleration noise, or roughness[1,9]. Thus, to identify acceptable noise levels, as well as to solve other motion cueing issues, the influence of background accelerations on subjects' sensitivity to their variation (differential thresholds) should be investigated.

Knowledge of perception of acceleration over-threshold values is useful for estimating the effect of accelerations caused by abrupt aircraft response to pilot control inputs (for small roll mode time constants) and for estimating motion fidelity in cases where acceleration noise exceeds the threshold level, etc[1]. It is known that the relationship between the intensity of a sensation and a physical stimulus (weight, sound, pressure, etc.) usually corresponds to the general psychophysical laws (Weber, Fechner, Stevens). Thus, it is worthwhile to check if these laws also hold true for accelerations.

In summary, the paper considers the following issues:

- the roles of different human sensory systems in acceleration perception and the corresponding math-models,
- absolute and differential acceleration perception threshold values, in particular the influence of specific forces on angular acceleration threshold values,
- the perception of over-threshold accelerations.

1. Description of experiments.

The experiments were conducted on TsAGI's Flight Simulator FS-102, which has a six-degrees-of-freedom synergistic motion platform. The motion system consists of 6 actuators with hydrostatic bearings. The actuator stroke is 1.8 m and the maximum values of displacement, velocity and acceleration for each degree of freedom are shown in Table 1.

Table 1

	Travel, *m , deg*	Velocity, *m/sec, deg/sec*	Acceleration, *m/sec^2, deg/sec^2*
Surge	±1.75	1.5	7
Sway	±1.23	1.1	8
Heave	±1.475	1.3	7
Roll	±35.1	30	230
Pitch	±37.8	30	230
Yaw	±60	50	260

The acceleration noise of FS-102 is typical of such simulators. For example, Fig.1 illustrates the acceleration noise in heave. This is so-called total noise determined in accordance with the AGARD[9] (rms of all but the first harmonic related to the amplitude of the first harmonic; the input accelerations were sinusoidal with a frequency of 0.5 Hz).

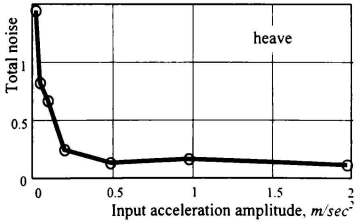


Fig.1 - Motion system acceleration noise in heave (f = 0.5 Hz).

The positions of subjects (head position) relative to the rotational axes were 1.6, 0 and 0.45 meters along vertical, lateral and longitudinal axes respectively.

To measure and register accelerations along all degrees of freedom, 6 acceleration transducers placed in the simulator platform were used.

In most publications experimental investigations of acceleration perception have been conducted using either step or sinusoidal input accelerations. In the present work sinusoidal accelerations were used. This particular technique has a number of methodological advantages as compared with step inputs since a) sinusoidal accelerations resemble separate components of accelerations in real flight, b) they are more accurately reproduced on a simulator and c) sinusoidal acceleration perception math-models are easy to identify. The technique is better suited to studying the roles of different sensory systems in motion perception because the dynamic characteristics of the different systems depend on acceleration frequencies.

Four subjects took part in the experiments. Each subject wore a helmet with a covered view to exclude visual perception and any perception of audible actuator noise to prevent a subject from identifying the frequency and direction of motion through other sensory channels.

In the majority of experiments the sensitivity thresholds were determined by the level of acceleration at which a subject started to identify the direction of motion ("identification level"). In the experiments on differential thresholds, threshold was defined to be the level at which a subject started to feel the presence of acceleration ("barely perceptible level").

The instant the sensations started or changed their nature, the subject registered by moving a stick. To check whether subjects were identifying motion direction correctly, the direction of input acceleration and stick deflection were compared.

For each subject from N=3 to 15 evaluations of acceleration threshold were performed for each configuration considered. The threshold values received for each subject were averaged.

2. The Roles of Different Sensory Systems in Acceleration Perception.

As mentioned above, the kinesthetic and tactile sensory systems participate in acceleration perception as well as the vestibular system. However, there are no experimental data on these sensory systems' roles. This can be accounted for by the fact that the acceleration perception process involves the integration of data from different sensory systems and, thus, it is almost impossible to study empirically the role of a single sensory system in this process (although it might be feasible in experiments with handicapped subjects with certain sensory disabilities). However, we discovered that pre-trained subjects have the ability to single out separate sensation components against the background of the other components, just as a pre-trained musician is able to single out the tones of different pitch from a musical chord. Taking this analogy a stage further, vestibular sensations resemble low-pitched tones, kinesthetic sensations resemble middle-pitched tones and tactile sensations resemble high-pitched tones. This fact enables us to study the dynamics and roles of the different sensory systems in acceleration perception.

In the experiments the simulator platform moved in sinusoidal mode along each degree of freedom. For example, in surge:

$$X = A(t)/\omega^2 \sin \omega t \qquad (1)$$

In every test the acceleration amplitude of sinewaves A increased slowly at a constant rate, the frequency being constant. The rate of change was quite small ($\dot{p}, \dot{q}, \dot{r}$ = 0.069 deg/sec^3, $\dot{n}_{x,y,z}$ = 0.0036 m/sec^3). The tests were repeated for different frequencies, from 0.5 to 8 rad/sec.

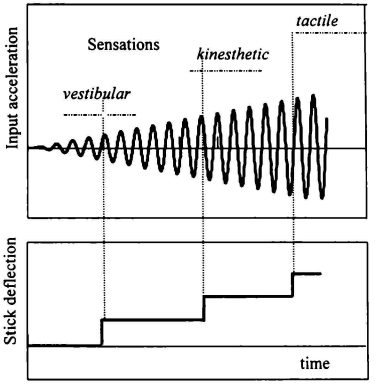


Fig.2 - The procedure of determining the boundaries of different sensation components.

The subjects' task was to analyse the nature of their sensations in order to identify the sensory system or the body part which was involved in motion perception (vestibular system sensations, displacements of the body or its parts due to inertia, etc.). On each occasion when the sensations started or changed their nature, the subject registered by applying a step input on the manipulator (fig.2) and by voice.

Table 2

Types of sensation	n_x	n_y	n_z	p	q	r
vestibular	+	+	+	+	+	+
kinesthetic (inertial):						
- torso	+	+	+	-	±	-
- certain parts of the body	+	+	+	-	-	-
tactile (skin-muscle):						
- pressure	+	-	+	-	-	-
- displacement	+	+	-	-	-	-
- high-frequency shifting	+	+	-	-	-	-

Notation: "+" - presence of sensations, "-" absence of sensations, "±" - negligible sensations

The results showed that the roles of sensory systems are different for different degrees of freedom. Table 2 presents subjects' evaluations of the sensory systems roles.

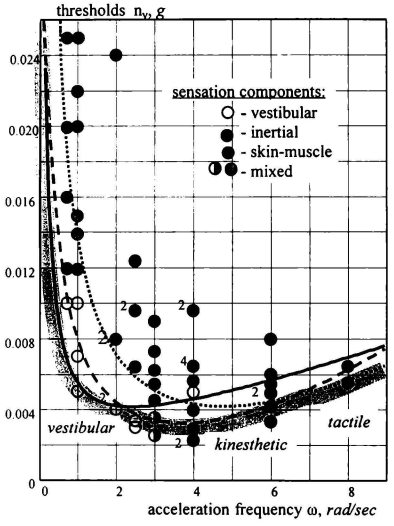


Fig.3 - Sensitivity boundaries of different sensory systems.

It can be seen that in angular motion perception the vestibular system dominates. In the perception of linear specific forces, kinesthetic and tactile sensors play important roles as well.

The roles of the various sensory systems also depend on frequency. Fig.3 shows the roles of different systems in the perception of lateral specific forces.

The boundaries presented in this figure are in fact the sensitivity thresholds for the vestibular, kinesthetic and tactile sensory systems respectively. We have also obtained boundaries for vertical and longitudinal specific. The results show that for frequencies up to 1.5-2 rad/sec the vestibular system dominates in creating acceleration sensations for all linear degrees of freedom but for higher frequencies kinesthetic and tactile systems dominate.

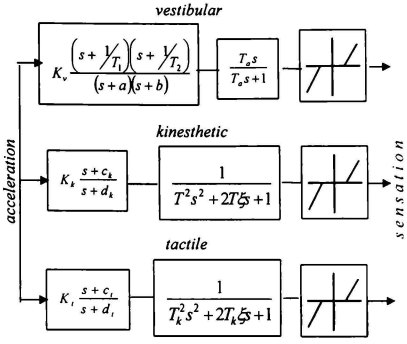


Fig.4 - Multi-sensory model of motion cues perception.

Based on these results and on published data on individual receptors, a multi-sensory model is proposed which takes into account the roles of different systems in acceleration perception (fig.4). The architecture of the model is the same for different degrees of freedom, though the values of its parameters differ.

This model and most of those available in publications were developed to describe the acceleration perception over a wide frequency range, including low frequencies. For the majority of motion cueing issues we are interested mainly in the 0.1-3Hz frequency range. First, this frequency range is typical of tracking and stabilization tasks where acceleration effects are significant. Second, due to simulator displacement and dynamic response limitations only this frequency range can be reproduced on ground-based simulators. Thus, for the majority of motion

cueing issues there is no need for a complicated math-model, and a model which describes the acceleration perception dynamics within a limited frequency range would be sufficient. Accuracy of solution of different simulation problems is determined not just by the accuracy of the acceleration perception mathematical description, but also by the accuracy of pilot-aircraft system analysis methods. At present these methods are more qualitative than precise. Thus, in practice, there is no need to know the exact acceleration perception dynamics even within the mentioned frequency range. Assuming this is true, the model can be simplified to the form shown in fig.5 for cases where the frequencies of accelerations and angular rates are within the 0.1 - 3Hz range.

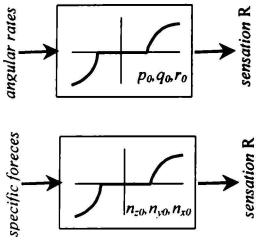


Fig.5 - Simplified model of motion cues perception for frequencies within 0.1-3 Hz.

We now wish to determine the values of the model's parameters, i.e. threshold values for p,q,r and $n_{x,y,z}$, and the perception laws for over-threshold values of accelerations $R(p,q,r)$, $R(n_x, n_y, n_z)$.

3. Sensitivity Thresholds.

There are some data on sensitivity thresholds in publications, but those data are incomplete. In our experiments the sensitivity thresholds were identified for all degrees of freedom using sinusoidal accelerations.

Absolute sensitivity thresholds. In the experiments the simulator moved in accordance with Eq.(1). In every test the amplitude of sinewaves increased slowly from zero until a subject was able to distinguish the direction of the motion (regardless of the sensory systems participating in perception). The corresponding acceleration amplitude was assumed to be the acceleration sensitivity threshold.

Experimental data, obtained for all degrees of freedom are presented in fig.6.

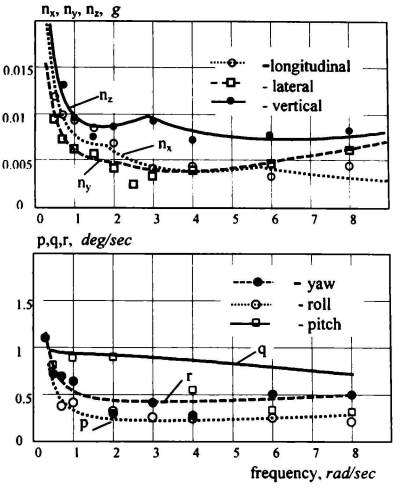


Fig.6 - Absolute sensitivity thresholds for linear and angular accelerations.

In the lower figure, angular sensitivity thresholds are shown. It can be seen that threshold values are about 0.4-0.9deg/sec depending on the degree of freedom. The angular threshold values are practically independent of frequency for the frequencies typical of piloting. This means, in particular, that the angular motion sensations are proportional to angular rate, not angular acceleration (this fact is used in the math-model, fig.5). There is a simple physical explanation for this. From Table 2 it follows that the vestibular system (semicircular canals) plays the dominant role in angular motion perception. Previous work[2-4] shows that within the frequency range from 0.1 to 10 rad/sec the semicircular canals function like integrating accelerometers.

In the upper figure, sensitivity thresholds for linear accelerations are shown. The threshold values are practically independent of the acceleration frequency for frequencies above 1.5rad/sec indicating that, for these frequencies, perception is a function of linear acceleration. (This is the basis of the acceleration perception math-model, fig.5.) The results in fig.6 show that operators seem to be very sensitive to lateral accelerations; threshold values are approximately

0.003g for frequencies typical of piloting. The maximum threshold values are in the vertical axis, where they are 0.008g. As frequencies decrease, threshold values increase because of the operator's adaptability to low-frequency accelerations.

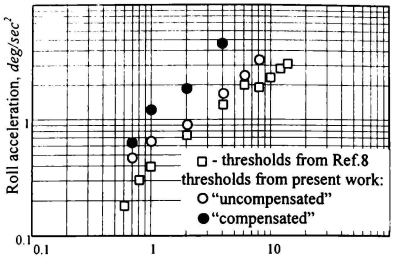


Fig.7 – Comparison of the thresholds obtained with those in Ref.8.

The results obtained are consistent with those in Ref.8 for roll, pitch and heave. Fig.7 presents a comparison of sensitivity thresholds for roll in terms of angular accelerations.

While investigating sensitivity to pitch and roll accelerations on the ground, it is not always possible to compensate for the longitudinal and lateral specific forces caused by cockpit tilting, since the simulator displacement capabilities are often low or absent (as in Ref.8, for example). Our experiments showed that the lateral specific force due to cockpit tilting has a considerable effect on roll sensitivity thresholds on the ground. The "compensated" thresholds are approximately 1.5-2 times greater than "uncompensated" thresholds (fig.7). Thus, data on roll sensitivity obtained in simulators without tilt compensation cannot be absolutely correct. The longitudinal specific forces due to cockpit tilting do not have a significant effect on pitch sensitivity thresholds.

Differential sensitivity thresholds. Differential thresholds were determined for specific forces for all three linear degrees of freedom. Such data are needed to identify acceptable levels of roughness when reproducing both specific forces and angular accelerations[1,9,10].

The background specific force was sinusoidal with frequency ω_0 = 1rad/sec. The amplitude of the background specific force A_0 was constant in each test but was varied from test to test. Subjects' sensitivity to the variation in specific force, $\Delta n(t)$, was determined.

The variation in specific force $\Delta n(t)$ was a high-frequency sinusoidal specific force with frequency ω_0 = 4 rad/sec:

$$\Delta n(t) = (aA_0 \sin \omega_0 t)\sin \omega t .$$

The total specific force was as follows:

$$n = A_0 \sin \omega_0 t + \Delta n(t) \qquad (2)$$

From fig.8 and Eq.(2) it is seen that a high-frequency specific force was reproduced against the background of low-frequency specific force.

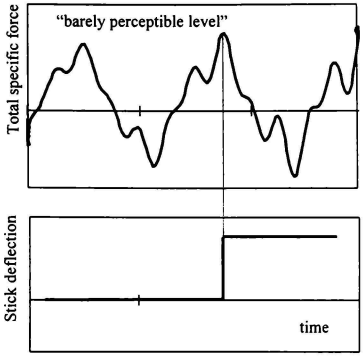


Fig.8 – The procedure of determining the differential thresholds.

The amplitude ($aA_0 \sin \omega_0 t$) of the high-frequency specific force was proportional to the current value of the low-frequency specific force. In each test the amplitude of imposed acceleration increased slowly (a=0.001, sec^{-1}) up to the moment when the subject was able to feel its presence against the background acceleration ("barely perceptible level").

Fig.9 shows the data for differential thresholds. It can be seen that as the amplitude of background acceleration increases, differential sensitivity thresholds increase linearly, regardless of the degree of freedom:

$$\Delta n = n_0 + kn, \qquad (3)$$

where n_0 is an absolute sensitivity threshold and k is a constant, different for different degrees of freedom.

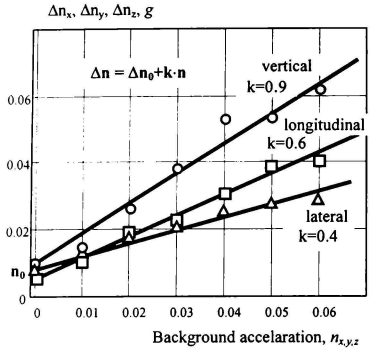


Fig.9 - Differential sensitivity thresholds vs background accelerations

The figure shows that differential thresholds are significant. Depending on the degree of freedom, differential threshold values vary from 40 to 90 % of the background acceleration amplitude.

Effect of specific forces on operators' sensitivity to angular motion. The experimental data available in publications refer only to perception in single degrees of freedom. However in real flight or on ground-based simulators specific forces and angular accelerations act simultaneously. If we are to understand the roles of motion cues in piloting, we therefore need to know the effect of specific forces on angular motion perception[1,10].

In the experiments, the effect of vertical acceleration on angular motion perception was considered. For practical purposes the effect of *Z*-axis specific forces on angular motion perception is the most important because maneuvers in the vertical plane are dominant in flight. As far as other specific forces are concerned, their effects are similar in nature and, thus, they may influence the angular sensitivity threshold values to a considerable degree.

Fig.10 shows the experimental data on pitch rate thresholds for various background normal acceleration amplitudes (normal acceleration frequency was 1rad/sec, pitch mode frequency was 4 rad/sec). It is seen that normal accelerations have a considerable effect on human operator sensitivity to angular motion. For example, as accelerations increase from 0 to 0.05g, angular sensitivity threshold values become 2-3 times greater.

The pitch rate threshold values depend on vertical acceleration values according to the linear function:

$$q = q_0(1 + 18A_{nz})$$

where q_0 is the absolute pitch sensitivity threshold and A_{nz} is the normal acceleration amplitude.

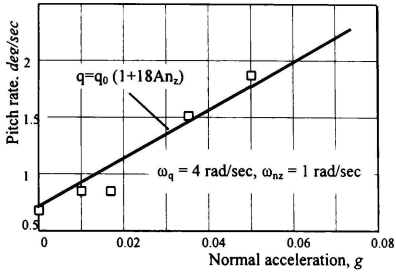


Fig.10 - Pitch thresholds as a function of vertical accelerations.

4. Over-threshold acceleration perception laws.

In the 19th century E.Weber discovered that the smallest noticeable difference in a stimulus ΔS was roughly proportional to the intensity of the stimulus S (it was mainly founded in experiments where subjects were given two nearly identical stimuli, for example, two similar weights), i.e.:

$$\frac{\Delta S}{S} = const \qquad (4)$$

Usually, ΔS is known as the differential sensitivity threshold if $S \neq 0$

It was suggested later that the function of sensations' intensity R of stimulus' value follows the Weber-Fechner law:

$$R = k \log S \qquad (5)$$

It was proved then that this law holds true only for medium intensity stimuli (i.e. it is not applicable where stimuli are very weak or very strong).

Since the 1950s, Stevens' formula has been used to define sensations for a wide range of stimulus intensity, excluding very strong stimuli:

$$R = k\,(S-S_0)^{\alpha}, \qquad (6)$$

where α, k are constants which depend on the type of stimuli and S_0 is the absolute sensitivity threshold.

Laws (4)-(6) are true for various stimuli, including tactile and kinesthetic sensations. As shown in section 2, these sensory systems play an important role in acceleration perception. Thus, it would be natural to expect these laws to hold true for accelerations as well.

The data on differential thresholds (Fig.9, Eq.3) prove the applicability of the Weber-Fechner law to accelerations. Eq.(3) at $n_y >> n_{y0}$ is, in fact, equivalent to Weber's law:

$$\Delta n = n_0 + kn \approx kn, \quad \text{or} \quad \Delta n/n = k.$$

Applicability of laws (5) and (6) to accelerations is also proved by the experimental data in fig.11.

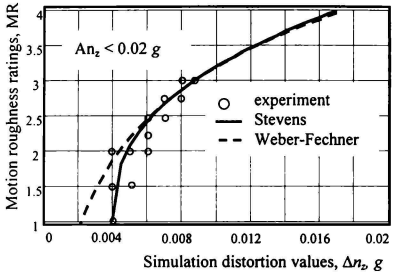


Fig.11 - Normal acceleration simulation ratings vs simulation distortion values.

The figure shows motion roughness (MR) ratings of the distortions while reproducing normal accelerations. The distortions were modelled as a sum of two sines with frequencies of 1 and 3 Hz. In Fig.11 Δn_z is the maximum value of the distortion. The MR ratings were estimated using a special scale which has four grades, MR=1 being the highest fidelity, when the distortions are not felt by a pilot, and MR=4 being the lowest fidelity, when the intensity of the jerks prevents aircraft motion being distinguished due to the distortions.

Fig.11 shows that acceleration perception can be adequately described by Stevens' law, which in our case (R=MR-1, S=n_z) has the following form:

$$MR = k(n - n_{z0})^{\alpha} + 1,$$

where k=17, α=0.4, n_{z0}=0.004g for the normal accelerations.

It can also be seen that, for acceleration values $n_z \geq 0.006g$, the relationship between motion fidelity ratings and acceleration noise values can be described by the following expression:

$$MR = 9.85 + 3.36\log(n_z),$$

This proves the applicability of the Weber-Fechner law to the accelerations of over-threshold values.

Conclusions

Experimental results on the roles played by different sensory systems in acceleration perception have been obtained for each of the six degrees of freedom. A multisensory math-model has been proposed to describe the acceleration perception within a wide frequency range and a simplified math-model has been proposed to describe acceleration perception within the typical piloting frequency range.

The paper has presented results on the absolute thresholds of sensitivity to sinusoidal accelerations for all degrees of freedom. It has shown that the lateral specific forces due to cockpit tilting produce a significant effect on roll sensitivity thresholds; results obtained in simulators without tilt compensation cannot be as accurate.

The paper has also presented results on differential sensitivity threshold values for specific forces. It has shown that differential threshold values are significant. Depending on the degree of freedom, the differential threshold values vary from 40 to 90% of the background acceleration.

Vertical accelerations have a considerable effect on human operator sensitivity to angular motion. For example, as accelerations increase from 0 to 0.05g, angular sensitivity threshold values become 2-3 times greater.

Finally, the paper has shown that the perception of over-threshold values of accelerations follows the general psychophysical laws (Weber-Fechner, Stevens).

References

1. A.D.White, V.V.Rodchenko. "Motion Fidelity Criteria Based on Human Perception and Performance". From AIAA Flight Simulation Technologies Conference, Portland, August, 1999.

2. A.A.J.van Egmond, J.J.Groen, L.B.W.Jongkees. "The Function of The Vestibular Organ". Practica-Oto-Rhino-Laryngologica, Vol.XIV, Supplement 2, 1952.
3. L.R.Young. "The Current Status of Vestibular models". Automatica, Vol.5, pp.368-383, 1969.
4. L.R.Young. "Vestibular System Models". Proceedings of International Symposium on Technical and Biological Control Problems, Erevan, USSR, 24-28 Sept., 1968, pp.93-119 (in Russian).
5. R.J.A.W.Hosman, J.C.van der Vaart. "Visual-Vestibular Interaction in Pilot's Perception of Aircraft or Simulator Motion". In: Essays on Stability and Control. Report LR-600, October 1989, TU Delft, The Netherlands.
6. J.L.Meiry. The Vestibular System and Human Dynamic Space Orientation. Ph.D.Thesis, MIT, June, 1965.
7. R.Lancraft, J.Zacharias, Baron. "Pilot Model Analysis of Visual and Motion Cue Requirements in Flight Simulation". AIAA, 1981, June 16-18, pp.49-59.
8. R.J.A.W. Hosman and J.C.van der Vaart. "Thresholds of Motion Perception Measured in a Flight Simulator." Proceeding of the twelve NASA-university annual conference on manual control, 1976, USA. p.956-983.
9. Working Group: "Dynamic Characteristics for Flight Simulator Motion Systems", AGARD Advisory Report No.144, AGARD, NATO, Neuilly sur Seine, France, September 1979.
10. V.V.Rodchenko, L.E.Zaichik, Yu.P.Yashin. "Motion Cueing on Research Flight Simulators". CEAS Symposium on Simulation Technology "Making it Real", Delft, The Netherlands, 30,31 Oct. - 1 Nov., 1995.

ADVANCED NUMERICAL TECHNIQUES FOR HI-FIDELITY ON- AND OFF-ROAD VEHICLE SIMULATION

Dr. Dario Solis
Modeling and Simulation, Chief
NADS & Simulation Center
2401 Oakdale Blvd.
Iowa City, Iowa 52242-5003

Dr. Edward Haug
Professor of Mechanical Engineering
University of Iowa
Iowa City, Iowa 52242-1000

Siddhartha Shome
Graduate Student
The University of Iowa
2401 Oakdale Blvd.
Iowa City, Iowa 52242-5003

Abstract

New developments in computer processor technology and in numerical techniques for integrating the equations of motion for vehicle systems have allowed increasing levels of fidelity in simulation, approaching the reality of the actual test track. High-fidelity vehicle models addressed include components such as steering, chassis, and suspension assemblies, modeled using the high fidelity multi-body dynamics approach. Other key vehicle subsystems such as power assisted steering, aerodynamics, braking, powertrain, and tires are modeled using the state space approach to support vehicle designers in industry in conducting virtual prototyping studies of vehicle components and complete vehicle systems. In the past, these simulation studies required long simulation runs that decrease the potential for using high-fidelity dynamic simulation as an interactive design tool. This paper presents recently developed numerical techniques in vehicle modeling, numerical integration, and parallel processing that significantly reduce the simulation time for complex vehicle systems. At the high-performance extreme, real-time vehicle dynamics simulation capabilities have been developed to support driver-in-the-loop simulation applications. Vehicle models and special simulation environments developed for real-time simulation are presented.

Introduction

The dynamic analysis of vehicle systems requires computer-based simulation tools with the ability of modeling vehicle chassis, and subsystem components. The most important chassis mechanical components are the vehicle frame, suspension assembly, and steering linkage assembly. Vehicle subsystems for the powertrain, steering, brakes, aerodynamics, and tire-soil need to be modeled in order to accurately represent the dynamic response of a complete vehicle system. Fidelity requirements in the modeling of each component are derived from each intended application. When approximate vehicle motion prediction is the goal, lower fidelity vehicle and subsystem components could be used. In this case, performance-based vehicle models that include six chassis degrees of freedom are developed using corresponding compliance coefficients determined either experimentally or by using a higher fidelity model. These simplified models very often include forcing functions with saturation elements for each vehicle subsystem to help the vehicle designer perform preliminary sizing of components and obtain approximate energy and power levels. After this stage is completed, the design synthesis process takes place. First the vehicle chassis component models are refined. During this process, the actual steering, suspension and frame assemblies are modeled by including all the physical bodies and force elements. Initially, a rigid multi-body approach is used and at later stages flexibility in joints, mounts, and bodies needs to be added. This process of continuos model refinement and physics based modeling enables the dynamic simulation process that directly generates dynamic loading information required for stress analysis and life prediction calculations. Later, this synthesis process is extended to the rest of the vehicle subsystems such as steering and powertrain.

In this paper, the structure of the equations of motion for hi-fidelity vehicle simulation will be first introduced to lead to the discussion of simulation and modeling techniques that allow for a more efficient numerical solution. A vehicle simulation example using both conventional and hybrid-electric powertrain designs will be used to illustrate some of the methods highlighted in the discussion. Conclusions and recommendations for future work are included at the end.

Multi-body Equations of Motion

The equations of motion for the vehicle chassis, suspension and steering assemblies, are generated using the constrained multi-body Newton-Euler equations. These equations are used in conjunction with the Lagrange multiplier theorem to construct a mathematical representation of the physical system. The use of the multi-body approach allows a direct mapping from computer aided design (CAD)-based representation of vehicle systems into the dynamic analysis space. Automatic tools for generating multi-body equations of motion have been available for a number of years[1]. These tools rely on libraries of basic constraint formulas, used to

represent kinematic joints, and libraries of force elements to construct the equations of motion. For a multi-body system model, the constraint vector equation can be expressed as

$$\Phi(\mathbf{r},\mathbf{p},t)=\mathbf{0} \qquad (1.1)$$

where $\mathbf{r}$ represents the vector of Cartesian displacements, $\mathbf{p}$ represents the vector of Euler parameters, and t denotes time. The Euler parameters for each body must satisfy the Euler parameter normalization constraint given by

$$\Phi^{\mathbf{p}}=\left[\mathbf{p}_1^T\mathbf{p}_1-1 \quad \ldots \quad \mathbf{p}_{nb}^T\mathbf{p}_{nb}-1\right]^T=\mathbf{0} \quad (1.2)$$

The variational form of the Newton-Euler equations in terms of virtual displacements and virtual rotations can be expressed as

$$\begin{aligned}&\delta\mathbf{r}^T\left[\mathbf{M}\ddot{\mathbf{r}}-\mathbf{F}^A\right]+\\&\delta\pi'^T\left[\mathbf{J}'\dot{\omega}'+\tilde{\omega}'\mathbf{J}'\omega'-\mathbf{n}'\right]=0\end{aligned} \qquad (1.3)$$

which must hold for all virtual displacements $\delta\mathbf{r}$ and virtual rotations $\delta\pi'$ that are consistent with constraints. This condition is expressed as follows

$$\Phi_{\mathbf{r}}\delta\mathbf{r}+\Phi_{\pi'}\delta\pi'=\mathbf{0} \qquad (1.4)$$

Since virtual rotations automatically satisfy Euler parameter constraint equations, these constraints were not included in Equation (1.4).

Based on the Lagrange multiplier theorem, there exists a vector λ such that

for arbitrary $\delta\mathbf{r}$ and $\delta\pi'$ the following equation holds

$$\begin{aligned}&\delta\mathbf{r}^T\left[\mathbf{M}\ddot{\mathbf{r}}-\mathbf{F}^A+\Phi_{\mathbf{r}}^T\lambda\right]\\&+\delta\pi'^T\left[\mathbf{J}'\dot{\omega}'+\tilde{\omega}'\mathbf{J}'\omega'-\mathbf{n}'+\Phi_{\pi}^T\lambda\right]=0\end{aligned} \qquad (1.5)$$

yielding the constrained Newton-Euler equations of motion

$$\mathbf{M}\ddot{\mathbf{r}}+\Phi_{\mathbf{r}}^T\lambda=\mathbf{F}^A \qquad (1.6)$$

$$\mathbf{J}'\dot{\omega}'+\Phi_{\pi}^T\lambda=\mathbf{n}'^A-\tilde{\omega}'\mathbf{J}'\omega' \qquad (1.7)$$

Equation (1.7) can also be written in terms of Euler parameters[1].

A number of different approaches is available in the literature[2] for the numerical solution of Equations (1.6) and (1.7) that satisfies Equation (1.1). A common approach is to reduce the set of differential algebraic equations (DAE) to a set of ordinary differential equations (ODE) on a manifold defined by Equation (1.1). This reduction can be implemented in many different ways, however, a common approach is to construct the null space of the constraint Jacobian matrix and use this basis to construct a set of independent coordinates that can effectively represent the dynamics of the system. The dimension of this independent set of equations is equal to the number of degrees of freedom of the system.

There is a number of factors that needs to be considered when choosing a solution method for the constrained multi-body equations of motion. Some of these factors are: (1) the degree of stiffness present in the physical system, (2) model dimension and topology structure, (3) need for computing reaction forces, and (4) presence of flexible elements.

In case stiff components are present in the physical system, implicit numerical integrators need to be used to efficiently integrate the equations of motion. These methods require effective means for calculating partial derivatives of the equations of motion with respect to generalized coordinates and generalized velocities. In this case, the use of joint space recursive methods geared towards reducing the dimension of the system of equations, increases the difficulty of computing sensitivities required for implicit integration[3] and optimization. This is due to the inherent increase in functional complexity during the reduction process.

The topology of the model allows obtaining significant performance gains in many different ways. Depending on the dimension of the problem, in the absence of significant stiffness, both direct Cartesian and joint space methods could exhibit reasonable performance. Several techniques based on the topology of the system can be applied to improve the performance of the solution process. For example, topology-based body and joint numbering schemes can be used to obtain coefficient matrix structures with attractive properties[4]. Similarly, topology based analysis methods have been devised for task scheduling for joint space recursive methods to exploit the parallel structure of the equations of motion[5]. If kinematic loops are not present in the problem, the application of joint space recursive techniques, results in an unconstrained reduced set of equations of motion. A null-space like method applied to the original set of equations of motion in Cartesian space, would still need to enforce the satisfaction of the kinematic constraint equations.

In searching for efficient ways of solving multi-body equations of motion, a basic trade off between the size of the system of equations and its complexity is encounter. For example, the dimension of Equations (1.6) and (1.7) could be reduced with the intention of minimizing the linear algebra computational load, however, the benefits deriving from using methods that exploit the sparsity of the original system of equations would be eliminated in the process. The extra cost of reducing the dimension of the problem represents a good investment if other performance improvements can be obtained, as it is the case with some parallel implementations.

When choosing a method, it is also important to consider the need for recovering joint reaction forces. Since there are methods that do not calculate the Lagrange multiplier vectors as part of the solution process, in the event that reaction forces are required, the extra effort to recover these forces a posteriori needs to be considered.

When flexible bodies are present, joined rigid- and flexible multi-body formulations are needed. In recent years, due to the need for improving the tire-road/soil interaction predicting capabilities, the simulation community has been more interested in combining rigid body and nonlinear flexible body tools. Several implementations of multi-body codes linked to finite element analysis (FEA) packages have appeared in the literature. These implementations are limited by the lack of freedom in modifying each application.

In order to illustrate some techniques to achieve better performance, a reduced-set of equations of motion containing kinematic loops will be used. These equations differ in dimension from the original set, however, the structure remains the same.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{T}\boldsymbol{\lambda} = \mathbf{Q}(\mathbf{q},\dot{\mathbf{q}},t) \qquad (1.8)$$

$$\mathbf{\Phi}(\mathbf{q},t) = \mathbf{0} \qquad (1.9)$$

Other than the reduced dimension, Equations (1.8) and (1.9) are solved using the same methods used to solve Equations (1.6) and (1.7).

In general there is the need for finding a suitable set of independent coordinates to reduce the original set equations and find a solution to system accelerations and velocities needed for numerical integration. The inherent difficulty in finding sets of independent coordinates, which is also applicable to joint space recursive methods when kinematic loops are present, is derived from the highly non-linear nature of general multi-body system equations. The functional relationship between the set of independent coordinates and the original set of generalized coordinates, is defined in terms of the position, velocity, and time. This means that methods that use fixed transformations between the independent set of coordinates and the original generalized coordinates, introduce artificial excitations that drive ODE integrators to reduce the integration stepsize. This parametrization dependent behavior is a result of a discretization in hyper-space of curvilinear spaces using rectangular coordinates. Efficiency gains during the simulation of vehicle systems lie in the generation, solution, and numerical integration of the equations of motion. These different potential performance gains will be addressed separately.

Generation of Equations of Motion

During the generation of the equations of motion, a number of terms need to be evaluated. From Equations (1.8) and (1.9), the inertia matrix $\mathbf{M}(\mathbf{q})$, the constraint Jacobian matrix $\mathbf{\Phi}_{\mathbf{q}}$, and generalized force vector $\mathbf{Q}(\mathbf{q},\dot{\mathbf{q}},t)$ need to be calculated. In vehicle dynamic simulation, a number of vehicle subsystems also need to be modeled. When components of the subsystem models are represented using the multi-body approach, such as is commonly the case with steering, and powertrain subsystems, the contribution to the chassis equations of motion is not only through the generalized force vector but also through inertia, and constraint forces. If the multi-body approach is not employed to model any part of a vehicle subsystem, chassis equations will only be affected through the external force vector. Figure 1 shows the topology structure of a 10-body independent suspension vehicle. In this picture, the steering mechanical components are modeled using a distance constraint to represent the connecting rods and a rack body attached to the chassis by means of a translational joint. In this particular model, no other subsystems are modeled using the multi-body approach. The rest of the subsystems are modeled as force elements following the state space approach. Those forces are included in the generalized force vector. More detail on these models will be given in a later section.

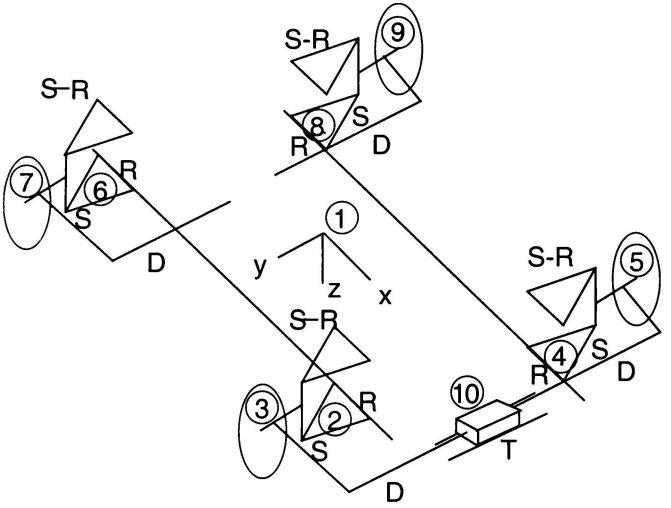


Figure 1. 10-body Vehicle Topology Model

Solution of the Equations of Motion

To solve the set of equations given by Equations (1.8) and (1.9) the differential null-space method will be used[6]. Instead of performing the coordinate transformation at the position level, it is performed at the velocity level. This transformation is given by

$$\dot{\mathbf{z}} = \mathbf{V}^T \dot{\mathbf{q}} \qquad (1.10)$$

The transformation matrix $\mathbf{V}$ is found by normalizing the non-singular matrix

$$\mathbf{P} = \begin{bmatrix} \Phi_{\mathbf{q}}^T & \mathbf{B}^T \end{bmatrix} \qquad (1.11)$$

Matrix $\mathbf{B}$ in Equation (1.11) is initially determined by decomposing the Jacobian matrix, using the SVD method, and completing the space to form the matrix $\mathbf{P}$. The process of normalizing the nonsingular matrix $\mathbf{P}$ to obtain the orthonormal matrix $\mathbf{V}$ is performed using the Gram-Schmidt orthonormalization process. Since $\mathbf{V}$ is orthonormal, Equation (1.10) is easily inverted to obtain

$$\dot{\mathbf{q}} = \mathbf{V}\dot{\mathbf{z}} \qquad (1.12)$$

The time derivative of Equation (1.12) is

$$\ddot{\mathbf{q}} = \mathbf{V}\ddot{\mathbf{z}} + \dot{\mathbf{V}}\dot{\mathbf{z}} \qquad (1.13)$$

where $\mathbf{V}$ is time dependent. The reduced equations of motion are obtained[8] by multiplying Equation (1.8) by $\mathbf{V}_I^T$ and using Equation (1.13) and (1.14).

$$\Phi_{\mathbf{q}} \mathbf{V}_I = \mathbf{0} \qquad (1.14)$$

The resulting equations of motion are given by

$$\mathbf{V}_I^T \mathbf{M} \mathbf{V}_I \ddot{\mathbf{z}}_I = -\mathbf{V}_I^T \mathbf{M} \dot{\mathbf{V}}_I \dot{\mathbf{z}}_I + \mathbf{V}_I^T \mathbf{Q} \qquad (1.15)$$

Equation (1.15) represents a set of ODE in the manifold defined by Equation (1.9).

Numerical Integration

The integration of the equations of motion given by Equations (1.15) represents the main challenge in hi-fidelity vehicle simulation. The combined set of DAE and ODE in the vehicle system of equations, need to be solved efficiently. Different stepsize requirements and stiffness levels require the application of dual-rate integration formulas[7], implicit integrators, and hybrid methods. In this paper a vehicle and subsystem simulation example is given using an ODE dual-rate integration approach with a null-space DAE solver[8]. Since vehicle models will be increasing in complexity and in the number of different disciplines that are included, hybrid integration environments

will be in more demand. The increasing number of issues that need to be investigated for the simulation of large hi-fidelity systems require a concentrated effort in the development of automatic techniques for identifying the structure of a given problem and provide a configuration for an efficient numerical environment.

Vehicle Subsystem Modeling

In order to reproduce the basic vehicle functionality, a number of basic subsystems need to be modeled. In this section, a discussion on the structure of basic models for steering, powertrain, and tire subsystems will be presented. The structure of these models is important since it impacts the type of performance that can be achieved in the overall simulation environment.

Steering System Model

The steering system presented here consists of the handwheel, steering column, rack and pinion, and steering linkages[9]. Power assisted systems vary depending on the type of power used, i.e. pneumatic or hydraulic. Depending on the level of complexity required and the simulation performance objectives (i.e., real time), a split model of the steering system can be developed. For example, in a rack and pinion implementation, the rack and connecting rods belong to the multi-body model while the rest of the components from the hand-wheel to the pinion, including the boost system, could be modeled using a state space approach. This type of separation is common since by including more mechanical parts of the steering system in the multi-body model, the simulation run times increase significantly. An easy way of linking the two components is to use a force-generating part interfaced with the multi-body system. This ODE-based module would receive inputs from the vehicle driver, have access to vehicle states, and be able to produce output forces as a function of internal state variables. This force is fed to the mechanical system during the generation of the equations of motion.
In the example given in this paper, a simple steady-state model of the steering system is created using a rotational spring-damper force-generating element. The equations below show the relative angular displacement and velocity used to calculate relative torques between the hand-wheel and the pinion.

$$\theta_{rel} = \psi - Nq_{rack} \qquad (1.16)$$

$$\dot{\theta}_{rel} = \dot{\psi} - N\dot{q}_{rack} \qquad (1.17)$$

$$F_{rack} = N(k\theta_{rel} + c\dot{\theta}_{rel}) \qquad (1.18)$$

Conventional Powertrain

The power-train subsystem model includes the prime mover (ICE or electric drive), torque converter, transmission, differential, and final gear assembly. In the case of hybrid electric power-train designs, batteries, controllers, generators, and energy management systems are also required. To interface the powertrain with the vehicle model the same split modeling approach is followed. Although, many of the components in the power-train can be modeled using the multi-body approach, for most applications, a split approach represents a better alternative. In this paper, the power-train is modeled as a torque generating dynamic function represented by internal state space variables, and driven by generalized coordinates and velocities from the multi-body system and inputs from the vehicle driver. Figure 2 shows a schematic of the computational flow in a four-wheel drive mechanical power-train model[10]. This model includes the following modules: engine, torque converter, transmission, transfer case, differentials, final gears and tires. A quasi-steady-state engine model is given as follows:

$$T_E = f_E\left(\dot{\theta}_E, \phi_f\right) \qquad (1.19)$$

$$J_E\ddot{\theta}_E = T_E - T_{Acc} - T_{TC_i} \qquad (1.20)$$

$$T_{Acc} = a_1 + a_2\dot{\theta}_E \qquad (1.21)$$

where:

$\dot{\theta}_E$ engine speed

ϕ_f governed throttle angle position

T_{Acc} engine accessory loss

T_{TC_i} torque converter absorbed torque

a_1, a_2 loss coefficients

As shown in Figure 2, the engine is connected to the transmission. This is done through a torque converter model. By using experimental data for the torque ratio γ and capacity factor K^2 as a function of the speed ratio curves both input and output torque converter torques can be estimated, as follows:

$$T_{TCi} = \frac{1}{K^2}\omega_{TCi}^2 \qquad (1.22)$$

$$T_{TCo} = \gamma\, T_{TCi} \qquad (1.23)$$

As noted above, the equations for the torque converter depend on the value of the rotational speeds on each side of the torque converter. In the

case of the output, the speed is calculated based on the speed at the interface between the torque converter and the transmission. This relationship is given by:

$$\omega_{TCo} = N_j \omega_{To} \qquad (1.24)$$

In a similar fashion, the transmission output torques are evaluated as:

$$T_{To} = \eta_j N_j T_{TCo} \qquad (1.25)$$

Where η_j represents the efficiency of the engaged gear sequence through the transmission and N_j represents the effective gear ratio. The transfer case model equation are given by:

$$\dot{\theta}_T = \left(\dot{\theta}_{FS} + \dot{\theta}_{RS}\right)/2 \qquad (1.26)$$

$$T_{FS} = T_{RS} = \frac{1}{2}\eta_T T_{To} \qquad (1.27)$$

$\dot{\theta}_T$ represents the angular velocity of the transfer case input shaft, $\dot{\theta}_{FS}$ and $\dot{\theta}_{RS}$ represent the front and rear output shaft speeds, respectively, η_T represents the transfer case efficiency and T_{FS} and T_{RS} are the evenly distributed front and rear shaft torques. An open differential model is used for both the front and rear. This model is given in terms of the following expressions

$$\dot{\theta}_{in} = \frac{N_D}{2}\left(\dot{\theta}_L + \dot{\theta}_R\right) \qquad (1.28)$$

where $\dot{\theta}_{in}$ represents the input shaft velocity, $\dot{\theta}_L$ and $\dot{\theta}_R$ represent the left and right axle velocities, and N_D represents the differential gear ratio. The torque ratio, α, is a function of the speed difference between the two axle shafts.

$$\alpha = 1 + \frac{1}{2}(\beta - 1)\left[1 - \cos\left(\pi \frac{\delta\dot{\theta} - \dot{\theta}_{\min}}{\theta_{\max} - \theta_{\min}}\right)\right] \qquad (1.29)$$

where β is the torque bias ratio, $\delta\dot{\theta}$ is the axle shafts' speed difference, $\dot{\theta}_{\min}$ is the minimum speed difference, $\theta_{\max}$ is the maximum angle difference between shafts, and $\theta_{\min}$ is the minimum angle difference between shafts. The torque output to the slower axle, T_{slow}, is expressed as:

$$T_{slow} = \eta_d N_D \frac{T_{in}}{1+\alpha} \qquad (1.30)$$

where η_d is the efficiency of the differential and T_{in} is the differential input torque. The torque to the faster axle, T_{fast}, is expressed as:

$$T_{fast} = \alpha T_{slow} \qquad (1.31)$$

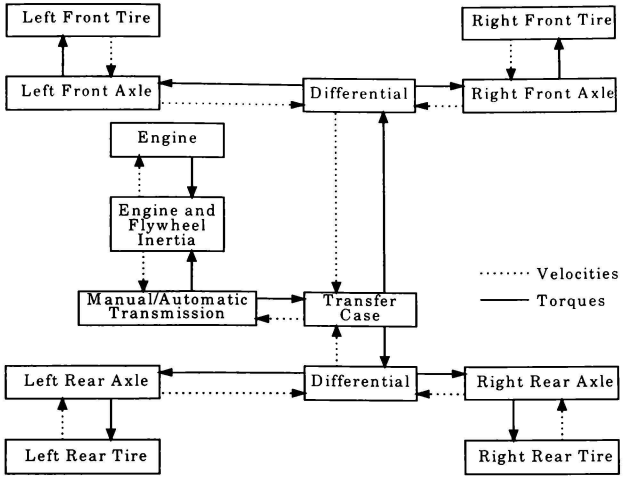


Figure 2. Four-wheel Driver Powertrain Computational Flow

Series Hybrid Electric Powertrain

An electrical powertrain model is described below.

v_{qs} quadrature stator voltage

v_{ds} direct stator voltage

v_{0s} zero axis stator voltage

v'_{qr} quadrature stator voltage referred to stator

v'_{dr} direct stator voltage referred to stator

v'_{0r} zero axis stator voltage referred to stator

ψ_{qs} quadrature stator flux linkage

ψ_{ds} direct stator flux linkage

ψ_{0s} zero axis stator flux linkage

ψ'_{qr} quadrature rotor flux linkage referred to stator

ψ'_{dr} direct rotor flux linkage referred to stator

ψ'_{0r} zero axis rotor flux linkage referred to stator

X_{ls} stator reactance

X'_{lr} rotor reactance referred to stator

X_{ss} stator self reactance

X'_{rr} rotor self reactance referred to stator

X_M mutual reactance

ω_b base speed of motor

ω_e electrical angular speed of stator voltage

ω_r rotor speed

ω rotational speed of rotor reference frame

A hybrid electric vehicle (HEV) powertrain is composed of several different components. The main power source is a battery bank, which supplies DC power to the system. An inverter converts direct current (DC) power to alternate current (AC) power. The AC power drives an induction motor that propels the vehicle. A secondary source of power is an internal combustion engine (ICE). This engine is coupled to a DC generator that recharges the battery bank.

The complexity of the component models varies depending on their effects on the overall simulation performance. The models of the ICE, battery, and inverter are rather simple or idealized, while those of the DC generator and induction motor are more detailed.

The induction motor is supplied three-phase voltage to its stator; and the resulting currents induce an electromotive force (EMF) on its rotor. The rotor begins to turn and accelerates until its rotational velocity is proportional to the electrical angular velocity of the AC voltage supply. The proportionality constant between the two is just the

number of poles of the motor. A computer model of an induction motor can be done in several ways, however, there are some choices that help simplify the calculations. The first choice is to use the electromagnetic fluxes in the motor as the state variables of the system instead of the corresponding currents. The units for flux used in this model are flux linkages per second. The second choice is to transform the stator and rotor equations to the ***dq0*** reference frame. The rotor reference frame will be allowed to rotate at some arbitrary speed. The right choice of reference frames will significantly reduce the frequency content in the state variables. The equations of motion in term of voltages and flux linkages[11] are given by

$$\left[v_{qs} \quad v_{ds} \quad v_{0s} \quad v'_{qr} \quad v'_{dr} \quad v'_{0r}\right]^T = \mathbf{G}\left[\psi_{qs} \quad \psi_{ds} \quad \psi_{0s} \quad \psi'_{qs} \quad \psi'_{ds} \quad \psi'_{0s}\right]^T \tag{1.32}$$

where $\mathbf{G}$ is given by

$$G=\begin{bmatrix} \frac{r_s X'_m}{D}+\frac{p}{\omega_b} & \frac{\omega}{\omega_b} & 0 & -\frac{r_s X_M}{D} & 0 & 0 \\ -\frac{\omega}{\omega_b} & \frac{r_s X'_m}{D}+\frac{p}{\omega_b} & 0 & 0 & -\frac{r_s X_M}{D} & 0 \\ 0 & 0 & \frac{r_s}{X_{ls}}+\frac{p}{\omega_b} & 0 & 0 & 0 \\ -\frac{r'_r X_M}{D} & 0 & 0 & \frac{r'_r X'_m}{D}+\frac{p}{\omega_b} & \frac{\omega-\omega_r}{\omega_b} & 0 \\ 0 & -\frac{r'_r X_M}{D} & 0 & -\frac{\omega-\omega_r}{\omega_b} & \frac{r'_r X'_m}{D}+\frac{p}{\omega_b} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{r'_r}{X'_{lr}}+\frac{p}{\omega_b} \end{bmatrix} \tag{1.33}$$

where

$$X_{ss} = X_{ls} + X_m$$

$$X'_{rr} = X'_{lr} + X_m \tag{1.34}$$

$$D = X_{ss} X'_{rr} - X_M^2$$

and p denotes the differentiation operator.

The speed at which the reference frame of the rotor equations rotates is determined by ω. The value used for ω is ω_e, the electrical angular velocity of the stator supply voltage. This is called a synchronously rotating reference frame. The electromotive torque developed on the rotor is given by the nonlinear equation

$$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)\frac{X_M}{D\omega_b}\left(\psi_{qs}\psi'_{dr} - \psi'_{qr}\psi_{ds}\right) \tag{1.35}$$

where P is the number of poles in the motor.

To increase the speed of the computer simulation, an order reduction of the induction motor model is implemented by eliminating the ***0s*** and ***0r*** quantities and neglecting the stator flux dynamics. These simplifications are reasonable if the voltages and loads in the three phases are balanced. The simplified equations[9] are given by

$$\begin{bmatrix} v_{qs} \\ v_{ds} \\ v'_{qr} \\ v'_{dr} \end{bmatrix} = \begin{bmatrix} \frac{r_s X'_m}{D} & \frac{\omega}{\omega_b} & -\frac{r_s X_M}{D} & 0 \\ -\frac{\omega}{\omega_b} & \frac{r_s X'_m}{D} & 0 & -\frac{r_s X_M}{D} \\ -\frac{r'_r X_M}{D} & 0 & \frac{r'_r X'_m}{D}+\frac{p}{\omega_b} & \frac{\omega-\omega_r}{\omega_b} \\ 0 & -\frac{r'_r X_M}{D} & -\frac{\omega-\omega_r}{\omega_b} & \frac{r'_r X'_m}{D}+\frac{p}{\omega_b} \end{bmatrix} \begin{bmatrix} \psi_{qs} \\ \psi_{ds} \\ \psi'_{qr} \\ \psi'_{dr} \end{bmatrix} \tag{1.36}$$

The order reduction to the induction motor model increased the simulation performance significantly with a small reduction in dynamic transient response.

Tire-Soil Interaction

The modeling of the tire soil interface can be broken down into three main parts. The first part is the modeling of the tire as a dynamic element. The second part is the modeling of the soil as material that interacts with the tires and the third part is the modeling of the interaction between the tire and the soil. Since real-time models are required to conduct VPG experiments in simulator environments, details about real-time tire modeling are also given.

The modeling of the tire is composed of two parts, the modeling of the rim and the modeling of the reinforced rubber. When a multi-body dynamic simulation tool with rigid and flexible body capabilities is used, a wheel rim is modeled as a rigid body and the reinforced rubber is modeled as a flexible body. The flexible body formulation could be based on either a modal approach or a full non-linear finite element approach. This distinction will dictate what type of data is required to feed the model. An environment like this is critical to develop an understanding of tire behavior at different frequency ranges. This high-fidelity vehicle simulation environment can be used to sort out what are the most critical elements in the tire modeling for both on- and off-road real-time applications. On road applications are characterized by hard soils. The attention then is shifted towards the contact between the tire and the hard soil or road.

When deformable soils are involved, commonly found in military, construction, and agricultural applications, predictions of soil deformation, motion resistance, and the development of traction are necessary. One approach is to develop finite element soil models supported by experimental research to guide the development of real-time soil models.

The vehicle-tire-soil system is implicit in nature. Loads from the vehicle to the ground and vice-versa depend on soil deformation and vehicle position and velocities. However, for hi-fidelity vehicle-tire-soil models, the resolution of the resulting set of nonlinear equations would be impractical. Therefore, explicit relations are utilized in different ways to provide a feasible solution. In on-road applications, the position and velocity of the rim are given explicitly. This information is used together with the fixed geometry terrain to determine the tire

deformation. In the case of finite element representations of the reinforced rubber, contact elements based on penalty methods identify the violation to the penetration constraint and find an associated force for each element. These forces are used to calculate the effective force and torque acting on the rim. Depending on the amount of elements used to model the tire, uneven load distribution at the contact patch can be reproduced.

After normal loads are computed, velocity information for each node at the contact patch is used to find frictional forces. Individual normal and tangential forces for each element are used to compute the moments in the three directions. At the end all forces and moments are added to calculate the effective values acting on the rim.

In the case of off-road applications, the explicit relationship between the rim and the tire is extended to the soil. Depending on the definition of the contact, the normal forces could be estimated based on the tire stiffness or soil stiffness. This approach is known as the master-slave method. In this method, the penetration gap is found and used in combination with the stiffness of the master to compute the reaction loads. Newton's third law is then used to apply a reaction to the slave that deforms based on its own stiffness.

The level of fidelity encountered when using combined rigid-flexible multi-body implementations for modeling the vehicle and the tire requires large amounts of computational power. These models are not intended for real-time applications. However, their availability is extremely helpful in the development of real time models.

Depending on the contact formulation used, normal loads are computed using contact geometry and tire and soil mechanical properties. For on-road applications the tire has been traditionally represented by a combination of springs and dampers. The simplest model used is that of a single spring and damper with an unstretched length which is compared to the distance from the tire attachment point to the point of contact. Some models query the terrain in a single point either right below the wheel attachment point or at a point defined by a tangent to the average contact patch. These single-spring-damper models are suitable for flat hard soils and very cheap to compute. After the normal load is calculated, kinematic information is used to determine longitudinal and lateral forces, and the self-aligning moment. Pacejka's magic formula is among the most common techniques for steady state calculations[12]. The need for higher fidelity tire models calls for a more complex contact formulation. One approach is to discretize the contact patch and perform multiple queries. This discretized contact patch is used to compute effective normal loads.

Example: Independent Suspension Vehicle System with Mechanical and HEV powertrain models

The example included in this paper contains the steering and powertrain models, both conventional and hybrid-electric, presented in earlier sections. The results shown correspond to a simple straight-ahead acceleration maneuver. The simulation runtime was 60 seconds. Figure 3, 4, and 5 show the position, velocity, and acceleration curves for both conventional and hev driven vehicles. The actual simulation times where 20 seconds for the mechanical and 40 seconds for the HEV. Both solutions were faster than real time.

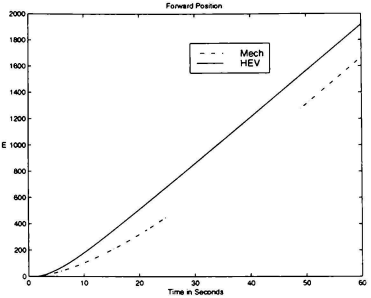


Figure 3. Vehicle Forward Position

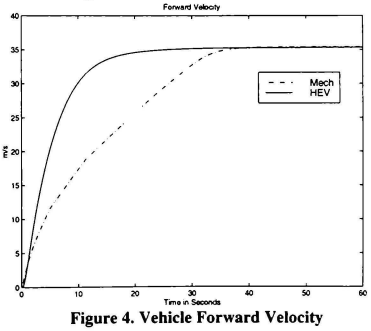


Figure 4. Vehicle Forward Velocity

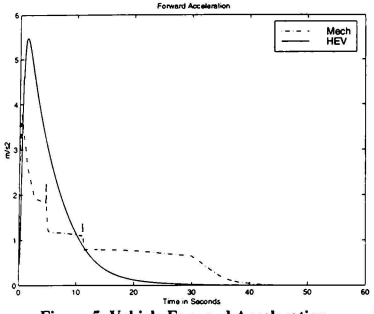


Figure 5. Vehicle Forward Acceleration

Conclusions and Recommendations

The efficient simulation of hi-fidelity vehicle systems calls for a deep understanding of the nature of the equations of motion of multi-body systems, physic based subsystem modeling, numerical integration, and computational methods. Even though computer power is increasing at a fast pace, the level of the modeling and simulation tools required for the applications of the 21st century has not been reached. More effort needs to be directed towards developing a common understanding of what modeling and simulation is and what is not. In this way the problems with the current capabilities can be identified and a plan for generating the solution and implementation for the future can be put in place.

[1] Haug, E.J., 1989, Computer-Aided Kinematics and Dynamics of Mechanical Systems, Volume I: Basic Methods, Allyn and Bacon.

[2] Hairer, E., and Wanner, G., 1991, Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems, Spring-Verlag.

[3] Haug, E. J., Negrut, D. and Iancu, M., "A State-Space Based Implicit Integration Algorithm for Differential-Algebraic Equations of Multibody Dynamics," Mechanics of Structures and Machines, Vol. 25, No. 3, 1997, pp. 311-334.

[4] Serban, R., Negrut, D., Potra, F. A. and Haug, E.J., "A Topology Based Approach for Exploiting Sparsity in Multibody Dynamics in Cartesian Formulation," Mechanics of Structures and Machines, Vol 25(3), 1997, pp. 379-396.

[5] Tsai, F.F., 1989, "Automated Methods for High Speed Simulation of Multibody Dynamic Systems," *Ph.D. Thesis, The University of Iowa.*

[6] Liang, C.G., and Lance, G.M., 1987, "A Differentiable Null Space Method for Constrained Dynamic Analysis," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, September, Vol. 109, pp. 405-411

[7] Solis, D., 1996, "Multirate Integration Methods for Constrained Systems with Interacting Subsystems," *Ph.D. Thesis, The University of Iowa.*

[8] Solis, D., and Schwarz, C.W., "Multi-Rate Integration in HEV Virtual Proving Grounds,"Proceedings, ASME Design Automation Conference, 1998.

[9] NADSdyna User Manual, The University of Iowa, Iowa City, IA., August, 1995

[10] Freeman, J.S., Velinsky, S. A., (1991) "Four-wheel-drive Powertrain Models for Real-time Simulation", Advanced Automotive Technologies, ASME DE-Vol. 40

[11] Krause, Paul C. 1986. Analysis of Electric Machinery McGraw-Hill

[12] Paceka, H.B., and Bakker, E., The Magic Tyre Model, Proceedings of the 1st International Colloquium on Tyre Models for Vehicle Dynamics Analysis, Delft, Netherlands, October 21-22, 1991

USE OF THE MEDIATOR DESIGN PATTERN IN THE LaSRS++ FRAMEWORK

Kevin Cunningham*

Unisys Corporation
NASA Langley Research Center
MS 169
Hampton, VA 23681

Abstract

Software design patterns are effective, efficient, established solutions to common software design problems. The Mediator Design Pattern is a particularly simple design pattern. It is used extensively in the NASA Langley Standard Real-Time Simulation in C++ (LaSRS++) Framework. The Mediator Design Pattern acts as an information "broker" between the components of a system. The use of the Mediator Design Pattern is a simple means by which re-usability, simplicity and testability of flight simulation software can be obtained.

Introduction

The goals of maintainability, extensibility and reliability are certainly common to all software development efforts. The ability to achieve these goals grows more critical as the scope and intended lifespan of software increase. The use of design patterns and object-oriented analysis, design and programming techniques to meet these goals is well established.[1-4]

The NASA Langley Standard Real-Time Simulation in C++ (LaSRS++) Framework[5] is a large scale collection of classes which are used to build various simulations. The framework software is intended to have a long lifespan, outliving any one research simulation project which it is used to develop. With this in mind, care was taken to design the software using patterns and object-oriented techniques.

There is much commonality in the way certain software components interact. The relationships between components result in patterns which are repeated over and over regardless of the system being modeled. Establishing efficient and effective mechanisms for these relationships is a problem which requires careful attention. Design patterns are evolved, yet simple solutions to these common problems in software development.[2] By using "tried and true" design patterns, a software developer is spared the time and expense of iterating to an efficient solution to the problem.

The Mediator Design Pattern is a common software design pattern that is used extensively in the LaSRS++ Framework. The Mediator is responsible for passing data between, and managing the behavior of, its aggregate components. It is a simple, yet evolved solution to several common software design problems. The use of the Mediator Design Pattern simplifies component interfaces and eliminates component interdependencies, leading to a more simple design. This results in enhanced re-usability and increased testability. The final product is software with superior reliability, maintainability and extensibility.

*Senior Member, AIAA

Copyright ©1999 by the author. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

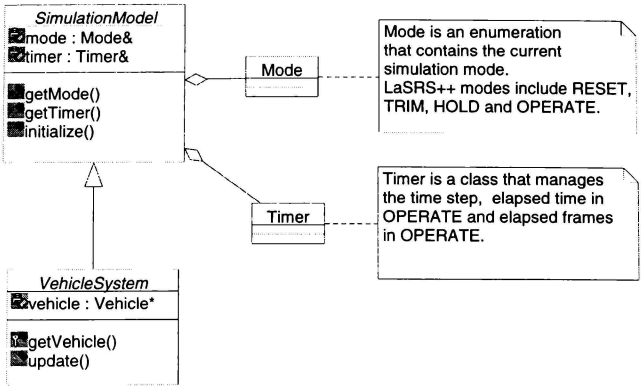


Figure 1: The Vehicle System

Fundamentals

Figure 1 is a class diagram[6] of the Vehicle System hierarchy. In the C++ language, a *class* is a software developer's fundamental building block. A single class represents a single concept or physical entity. It contains functions, which define behavior, as well as data. A particular instance of a class is called an *object*. Objects of various class types can be used to create complex software in a straight forward manner.

Two important relationships, *inheritance* and *containment*,[7,8] are illustrated in figure 1. Inheritance is also called an "is a" relationship (an F/A-18 is an airplane). It is the means by which the detail can be built up in a layered fashion. Containment is called a "has a" relationship (an F/A-18 has a flight control system). Containment occurs when one object instantiates (or contains a reference to) another object.

Figure 1 illustrates that the Simulation Model class contains Mode and Timer objects by reference. The variables which represent the references are `mode` and `timer`. The class also contains functions which a client can call to obtain access to the mode and timer objects. Additionally, the class contains a function defining behavior at initialization. The Vehicle System class inherits from the Simulation Model class. The vehicle pointer in Vehicle System is a pointer to the parent vehicle which contains the Vehicle System object.

A *software framework* is a collection of components which are used to build a variety products. In the case of the LaSRS++ framework, the products are simulations. One of the goals for the LaSRS++ framework is to foster good design at the aircraft model level via design choices implemented at the framework level. The Vehicle System concept is one example of this.

The design of the LaSRS++ framework is such that classes which are Vehicle Systems act as mediators. These mediator classes are intended to manage communication between the vehicle itself and its systems' components. The mediator is responsible for passing

data into its aggregates, instructing the aggregates to perform their calculations, and extracting any output. The Vehicle System is the framework's design pattern for decoupling a model from the vehicle.

Framework Systems

The Vehicle System class shown in figure 1 is an *abstract class*. Abstract classes are not designed to support the creation of objects, therefore there can be no instances of an abstract class. They can only be inherited. Abstract classes serve as a portal to system extension, providing the form which other classes are to follow. When inheritance relationships are chained together additional detail is added with each layer. The layer below the Vehicle System class provides common interfaces which in turn will be inherited.

The LaSRS++ framework provides support for a number of systems:

- Aerodynamic System
- Caution And Warning System
- Control System
- Fuel System
- Hydraulic System
- Landing Gear System
- Navigation System
- Propulsion System
- Weapon System

The class diagram for the some systems in the framework is shown in figure 2.

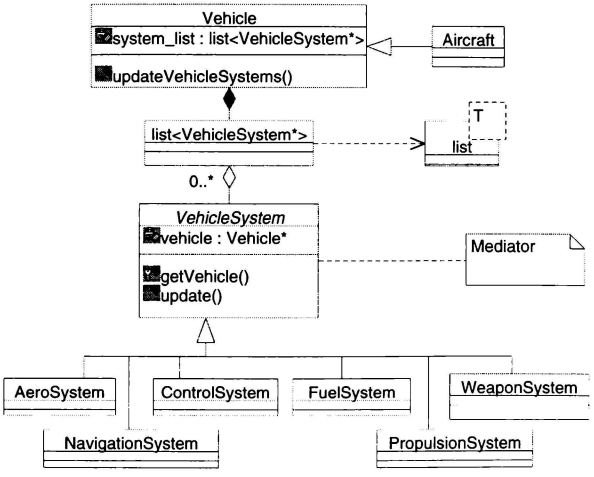


Figure 2: Systems in the Framework

Figure 2 shows the relationship between the Vehicle class and the Vehicle Systems class. A Vehicle has a linked list[9] of Vehicle Systems. There are no restrictions on the number of systems that can go on the list. As a specific aircraft instantiates its systems, the objects need only be added to the list. This registration is a simple matter of a single call to a function which is inherited from the Vehicle object. This makes it possible to update all the systems in a vehicle by simply sending one instruction to update whatever is on the list.

The system specific classes shown in figure 2 , which inherit from Vehicle System, establish the common interface that various simulations will use. When a specific vehicle simulation is being developed, the details specific to that simulation model are added to the interface. All systems of that type share the same interface, only the encapsulated details will vary.

A Closer Look At A System

Figure 3 is a more detailed class diagram. It shows how a mediator class fits into an architecture to decouple one part of a design from another. This means that a class representing a system being modeled does not require direct knowledge of the aircraft of which it is a part. The model is not dependent on the vehicle to perform its function.

Figure 4 is an object interaction diagram. The interactions depicted in that diagram are for a greatly simplified aerodynamic model. Enough detail is provided to illustrate how a typical mediator is used to decouple classes.

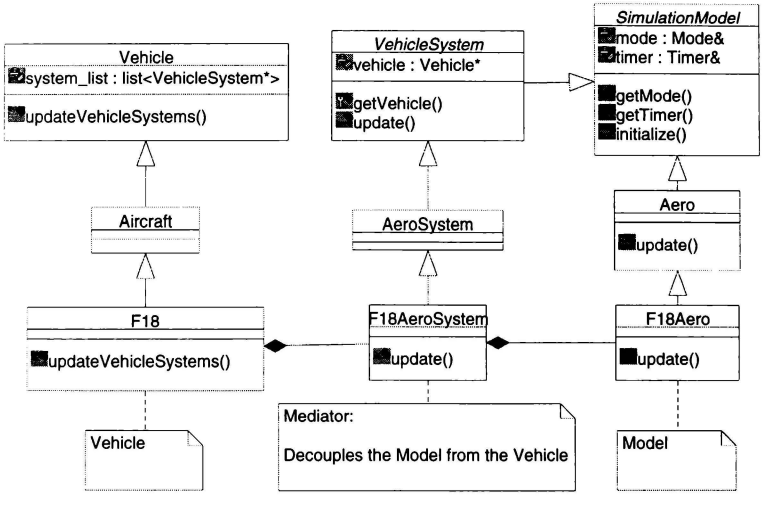


Figure 3: An Aerodynamic System Mediator

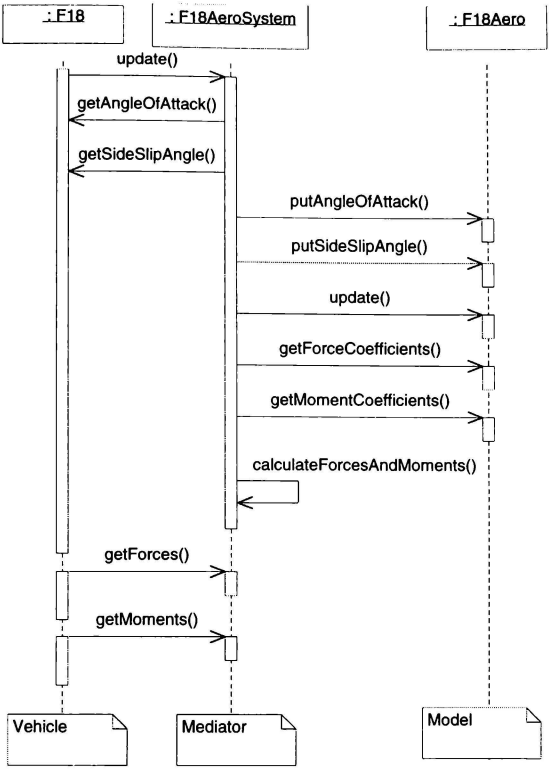


Figure 4: Aerodynamic System Object Interaction Diagram

Figure 4 shows that the interactions begin with the vehicle calling the update function in the mediator. The instruction (or message) for the Aerodynamic System to update itself triggers a series events. The mediator calls functions in the vehicle to obtain independent data required for table lookups in the aerodynamic model. After the necessary data has been obtained by the mediator, it is passed into the model. This occurs by calling publicly accessible inlined mutator functions which are defined in the model class. Mutator functions are the mechanism by which an object's internal, private data may be set. Next, the mediator instructs the aerody-

namic model to update itself. After the coefficients are calculated and totaled, the results are retrieved when the mediator calls the inlined accessor functions. Finally, the mediator calls a function defined within the mediator class which converts the coefficients into dimensional forces and moments.

The use of the mediator pattern allows the model class to remain blissfully ignorant of the vehicle's details. The mediator now contains the details required to manage the communication between the two objects. Complexity is shifted out of the model and into the mediator. In addition to decoupling components, the overall system is simplified by this design choice.

Major Benefits

Two major benefits of the decoupling provided by the use of the mediator pattern are increased testability and increased re-usability. Testing is fundamental to ensuring software validity and reliability. Prior to integration testing, it is desirable to test all of a model's components in an isolated environment. The key element to this type of unit testing is the ability to decouple the model from the rest of the simulation. Effort can then be put into developing highly effective test scenarios, rather than conjuring-up unique ways to isolate the test process from the simulation environment.

If "time is money", then the ability to re-use software is gold. A component with many dependencies cannot be re-used. Inter-component dependencies tend to make a large system stiff, in-flexible and difficult to extend. When multi-component interaction and behavior are mediated, the design of a single component readily supports re-use by different models, systems, simulations and facilities.

Lessons Learned

Figure 5 is the class diagram for the model of an aircraft's flight control law. It is a large and complex model. The figure illustrates the interdependencies that result when components of a flight control system share data through direct interaction. Each class is dependent on all the other classes from which it needs data. This creates a tightly coupled system. Any new data or behavior added to one class affects all the other classes that depend on it. As a tightly coupled system grows, it tends to take on the characteristics of a monolithic class. This limits re-usability, hampers testability and significantly increases the time required to compile the software. In a worst case scenario, every class would be dependent on every other class. *Don't let this happen to you!*

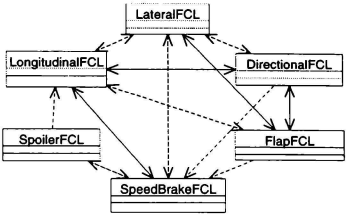


Figure 5: Class Dependencies Abound

The solution to this common design problem is the Mediator Design Pattern. The use of the Mediator Design Pattern rids objects of their explicit dependencies. This greatly decouples and simplifies a system. Figure 6 illustrates the impact of a mediator on the system shown in Figure 5.

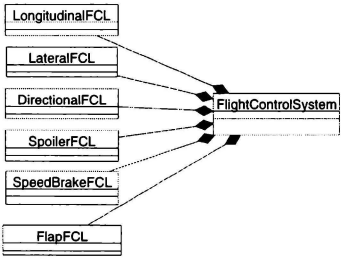


Figure 6: The Mediator Pattern To The Rescue

Figure 6 shows the more simple design. It is the superior design. The aggregate classes no longer depend on each other. In fact, they do not even depend on the mediator class which encapsulates them. This autonomy greatly increases re-usability. Testability is increased in that each class may be tested as a single unit, rather than testing the whole system at once. Finally, compilation times are reduced by the fact that a software change which only affects one class will only require re-compilation of that class, not an entire coupled system.

Conclusions

The NASA Langley Standard Real-Time Simulation in C++ (LaSRS++) Framework and aircraft simulations developed within the framework have successfully used the Mediator Design Pattern. The Mediator Design Pattern is a simple, effective, efficient, established solution to a number of common software design problems. The use of the Mediator Design Pattern results in software that is more easily tested at the component level, thus promoting a higher quality, more reliable product. The Mediator Pattern provides object decoupling, minimizing component interdependencies. This results in more simple designs and software which is more maintainable and extensible. By minimizing system coupling, the Mediator Design Pattern facilitates software re-use at not only the model and simulation levels, but also at the inter-facility level.

Bibliography

[1] Grady Booch. *Object-Oriented Analysis and Design*. Benjamin/Cummings, Redwood City, California, 1994.

[2] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.

[3] John Lakos. *Large-Scale C++ Software Design*. Addison-Wesley, Reading, Massachusetts, 1996.

[4] Robert C. Martin. *Designing Object-Oriented C++ Applications Using The Booch Method*. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[5] Richard A. Leslie, et al. LaSRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft. Paper Number AIAA-98-4529, August, 1998.

[6] Terry Quatrani. *Visual Modeling With Rational Rose and UML*. Addison Wesley, Reading, Massachusetts, 1998.

[7] Bruce Eckel. *Thinking in C++*. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[8] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Publishing Company, Reading, Massachusetts, third edition, 1997.

[9] Atul Saini David R. Musser. *STL Tutorial and Reference Guid*. Addison-Wesley, Reading, Massachusetts, 1996.

[10] Scott Meyers. *Effective C++*. Addison-Wesley, Reading, Massachusetts, second edition, 1998.

[11] Patricia Glaab, et al. A Method to Interface Auto-Generated Code into an Object-Oriented Simulation. Paper Number AIAA-98-4531, August, 1998.

[12] P. Sean Kenney, et al. Using Abstraction to Isolate Hardware in an Object-Oriented Simulation. Paper Number AIAA-98-4533, August, 1998.

[13] Scott Meyers. *More Effective C++*. Addison-Wesley, Reading, Massachusetts, 1996.

AIAA-99-4338

THE USE OF MULTIPLE THREADS IN AN OBJECT-ORIENTED REAL-TIME SIMULATION

David W. Geyer

Unisys Corporation
NASA Langley Research Center
MS 169
Hampton, Virginia 23681

Abstract

As real-time simulations become more complex, a single processor is no longer sufficient to perform all of the necessary computations. It becomes desirable to execute independent sections of a real-time simulation on different processors. The traditional approach to utilizing multiple processors on a symmetric multiprocessor machine involves writing a program composed of one *process* per processor that interact through shared memory. A program that uses *threads* can utilize multiple processors without the overhead of full processes or the use of a cumbersome shared memory mechanism. This is due to the fact that threads *implicitly* share a common address space. A multithreaded real-time program is organized as a main thread controlling zero or more auxiliary real-time threads. Each processor to be used for real-time computations is assigned a single auxiliary real-time thread. The main thread monitors the real-time clock and informs the auxiliary threads when the clock ticks. An auxiliary thread can also be used to monitor and control the simulation via a graphical user interface (GUI). These object-oriented designs were implemented in C++ for the NASA Langley Standard Real-Time Simulation (LaSRS++) Application Framework.

Introduction

Real-time simulations are continuous simulations that have a strict and direct correspondence between simulation time and real-world time: one unit of time in the simulation is equal to one unit of time in the real world. This type of simulation is usually used to represent or study a real dynamic phenomenon as it occurs. Real-time control of dynamic processes requires a human-in-the-loop or simulated controller (e.g, a pilot flying a flight simulator). Real-time simulations often involve modeling the evolution of continuous processes over time. This is accomplished by dividing time into small fixed intervals and solving differential equations at each successive interval of time. This type of computation is very time critical. It must be possible to perform all of the necessary computations in the allotted amount of time.

As real-time simulations become more complex, a single processor is no longer sufficient to perform all of the necessary computations. It becomes desirable to execute independent sections of the simulation model on different processors. The traditional approach to utilizing multiple processors on a symmetric multiprocessor machine involves writing a program composed of one *process* per processor that interact through shared memory. This involves using an explicit shared memory mechanism that must be managed by the program. This approach also presents a problem when using C++ [7] virtual functions: virtual functions for a given object are only available in the process that *created* the object (i.e., the virtual function table is bound to the address space of the creator process).

A program that uses *threads* can utilize multiple processors without the overhead of full processes or the use of a cumbersome shared memory mechanism. This is

Copyright ©1999 by the author. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

due to the fact that threads *implicitly* share a common address space. Because the virtual function tables are bound to the shared address space, C++ virtual functions for a given object are available in *any* thread.

Real-Time Simulation

In order to perform real-time simulations, it is necessary to synchronize the simulation execution with real-world time. This is normally accomplished by system level software that interacts with an external clock. The external clock must be accurate and have a precision on the order of microseconds or even nanoseconds. The system level software divides computer operation into fixed, equal intervals of time called *frames*. The simulation code waits for a frame to start, performs its computations, and informs the system level code when it is done. The simulation time is incremented by one frame time and the cycle repeats. As long as the simulation code takes less time to execute than a single frame, the simulation will continue to operate in real-time.

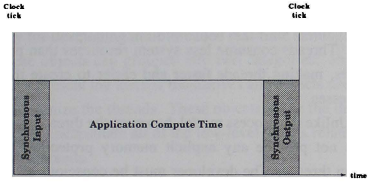


Figure 1: real-time frame

Figure 1 illustrates the concept of a frame. The synchronous input period of the frame is a variable length of time where information from the outside world is deposited into the memory space of the application code. Similarly, during the synchronous output period, application information is transferred to the outside world. These periods are considered synchronous since their execution is dependent on the system clock for start (input) and stop (output) times. The actual code that simulates the dynamic system runs during the application compute time.

If, for some reason, the dynamic system code takes longer to execute than the allotted application compute time, the simulation will have a deterministic behavior based on the nature of the real-time constraint. A *hard* real-time system will halt with a critical error if the real-time constraint is violated. A *soft* real-time system will allow computations longer than a frame and take some appropriate action without halting. Flight simulations that involve a human-in-the-loop control mechanism are hard real-time systems.

Process Model

The *process model* is an abstraction used by traditional operating systems to handle program execution. Each process is composed of an addressable memory space, a *single* sequence of machine instructions to execute, and other resources maintained by the operating system. Each process has its own, private address space, isolated from all other processes in the system. The process model was specifically designed in this way to allow programmers to write applications in isolation, unconcerned with the memory usage of other executing programs.

In a multiprogramming environment, several processes can be running on the system at the same time. Butenhof [1] defines *concurrency* as "things that appear to happen at the same time, but which may occur serially." On a uniprocessor system, processes only appear to execute concurrently. In reality, the operating system is providing each executing process a small amount of execution time on the CPU. Once this time expires, the operating system switches to another process to give it some time. This switching occurs among all running processes to allow them all to progress in execution.

A multiprocessor system not only provides concurrency, it can also provide true *parallelism*. Parallelism occurs when multiple processes are executing simultaneously on different processors. A parallel program can actually perform multiple tasks at once, while a concurrent program can only provide the illusion of multiple task execution.

In order to achieve true parallelism on a multiprocessor system with the process model, a program must be designed as a group of distinct processes that cooperate and share resources through some form of interprocess communication, such as shared memory. Most modern operating systems provide mechanisms to map, or overlay, a portion of a given process's memory space with the memory space from another process. Both processes

view the shared memory space as part of their own addressable memory space. As a result, any changes to the contents of the shared memory space by one process are visible to the other process.

Real-time vs. Time-sharing Processes

In a general purpose (a.k.a. non-real-time) operating system, the time-sharing process scheduler tries to ensure that all runnable processes will eventually get a chance to run. An I/O bound process will occasionally block and allow some other process to assume control of the CPU. A compute-bound process, on the other hand, will run until it is preempted by the scheduler. This is usually accomplished by periodically interrupting the running process on each CPU and determining if there is a higher priority process that should be run on that particular CPU. Each process is usually given a guaranteed amount to time that it can run on a CPU without being preempted. This period of time is called a *time slice*. At the end of each time slice, the scheduler will choose another process to run on the given CPU. The choice concerning which process to run next is made based upon a priority value that is associated with each process. As a process runs, its priority diminishes or degrades. When the priority of a process degrades to some minimum value, the priority will be restored to a maximum value.

In order to meet the strict timing requirements for real-time simulation, the traditional time-sharing scheduler must be bypassed. The operating system kernel must provide a mechanism to allow normal processes to be scheduled in real-time mode. A real-time simulation will require one or more actual processors to run in real-time mode.

Running a processor in real-time mode consists of:

- remove the processor from workload consideration by the normal, time-sharing process scheduler
- the ability to assign specific processes to specific real-time processors
- prevent the processor from servicing any interrupts that can be handled by other non-real-time processors
- prevent the preemptive scheduling on the processor; i.e., a process running on a real-time processor gives up control of the processor only when it chooses to relinquish control

Thread Model

The term *thread* refers to a single point of control that executes a sequence of instructions. The process model can be thought of as a specialization of the thread model where there is only a single thread. The *thread model* expands on the process model by providing the same abstraction as the process model but allowing *multiple* execution sequences. All of the process resources are shared by all of the threads.

The implicit sharing of process resources in a multithreaded program has benefits and drawbacks. The immediate benefit is the elimination of the need to use an explicit shared memory mechanism to communicate between threads. Another benefit is that a multithreaded program is a single program, whereas a multiprocess program is several separate programs that must work together. A single program is simpler to design and maintain. Threads consume less system resources than processes, making threads faster and easier to create than processes.

Unlike the process model, however, the thread model does not provide any explicit memory protection between threads. The developer must be conscious of the interaction of different threads on the common address space. If more than one thread can read and/or write a given data object, then some form of synchronization must be used to avoid data corruption. It is also possible for a given thread to corrupt the stack or heap memory used by another thread.

User Threads vs. Kernel Threads

There are two types of thread models available: user threads and kernel threads. User threads allow the developer to use threads as an organizational tool within programs, but do not allow for true parallelism. These types of threads are only visible within the given process; i.e, they are not visible to the operating system. As a result, the operating system cannot schedule these threads to run in parallel on multiple processors. The application itself has to schedule and manage the threads. As long as a

user threads application uses non-blocking system calls, such an application can maintain several independently executing tasks. A blocking system call will suspend the calling process until the system call completes.

Kernel threads, on the other hand, are recognized and managed by the operating system. They can be run in parallel on a multiprocessor machine. Kernel thread applications can also use blocking system calls, since the operating system will replace any blocked threads with a runnable thread. A kernel thread is usually implemented as two-part object. The *user-mode* portion is the application interface and data to the thread. The second part is the *kernel-mode* portion that exists inside the operating system kernel. The kernel-mode part is used by the operating system to schedule the thread as an independent entity. The kernel-mode portion also allows the operating system to bind the thread to one or more specified processors.

Object-Oriented Approach to Threads

A number of different types of objects are necessary for developing multithreaded real-time simulations. These objects can grouped into two categories: objects that represent the threads themselves and objects used to synchronize the threads. These objects allow the developer to encapsulate all of the low-level details related to using threads.

The C++ language does not provide any direct support for multithreaded programming. The approach used by LaSRS++ [3] is to use C++ wrapper classes around existing C interface libraries. LaSRS++ uses classes that encapsulate threads, mutexes and barriers. These classes hide all the low-level details of the actual thread package in use. Currently, LaSRS++ only supplies wrappers for SGI Irix share group processes [8]. Preliminary wrappers are in development for the POSIX threads package [1].

Thread Objects

A single wrapper class is used to represent a thread. This class contains all of the low-level data and function calls necessary to create, identify and destroy the thread. Since the mentioned thread libraries use a C language interface, the user must provide a C function that the thread will start executing after creation is complete. The developer must provide the wrapper object with the function as a constructor argument. Another constructor argument provides the option of allowing the new thread to start executing immediately, or of manually starting the thread at a later time.

Mutexes (or Locks)

The term *mutex* (short for mutual exclusion), refers to a synchronization primitive used to provide exclusive access to some shared resource. When a given thread wants exclusive access to a given shared resource, the thread tries to *lock* the mutex. If no other threads currently have exclusive access, the thread *acquires* the mutex and has exclusive access to the resource. If another thread already has exclusive access to the resource, the original thread will block (suspend) until the resource is *released* when the controlling thread *unlocks* the mutex. The mutex is said to be owned by the thread that has successfully locked the mutex.

Barriers

A *barrier* is a synchronization primitive used to bring a number of threads together during program execution. A barrier is initialized with the number of threads that it is required to synchronize, call this number n. Then, the barrier will suspend all threads that enter it until n threads are suspended in the barrier. Once the barrier contains n threads, all the suspended threads are allowed to continue.

Threads in a Real-Time Simulation

Kernel threads are required in order to build a multithreaded, real-time simulation. In addition, the operating system must provide the ability to bind specific threads to specific processors. A *real-time thread* is defined to be a kernel thread that has been bound to a processor running in real-time mode. Each processor to be used for real-time computations is assigned a single real-time thread. Each real-time thread assumes complete control of its assigned processor, until the thread decides to relinquish control.

A multithreaded, real-time program is organized as a main real-time thread controlling zero or more auxiliary real-time threads. This is often referred to as the "Work-Crew Model" [1]. The main thread monitors the

real-time clock and informs the auxiliary threads when the clock ticks; i.e., frame start. The auxiliary threads wait for frame start notification from the main thread, perform all computations for the given frame, and notify the main thread when computations are complete. Note that both the main and all auxiliary threads do not make any blocking system calls. An undeterministic duration system could cause a hard real-time simulation to miss a frame deadline.

Figure 2 illustrates the use of kernel threads in a real-time simulation. This approach is known as the one-to-one method [1]; i.e., one kernel thread to one processor. This approach is common for compute-bound threads, where blocking is not an issue.

Figure 3 shows the LaSRS++ classes that correspond to the main and auxiliary threads. A `MainSimulationThread` object contains a variable sized vector of `AuxiliarySimulationThread` objects. Every frame, the `MainSimulationThread` object will coordinate with the attached set of `AuxiliarySimulationThread` objects. Figure 4 shows the runtime interaction between a `MainSimulationThread` object and an `AuxiliarySimulationThread` object.

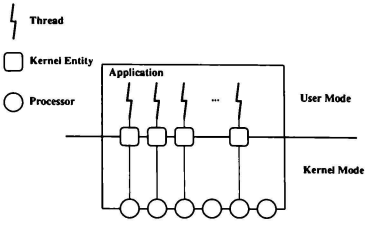


Figure 2: Real-Time Thread Allocation

The easiest way to partition computations between threads is to compute disjoint sections of math models on different threads. This eliminates the need for using time-consuming thread synchronization primitives during the compute phase of the frame. If thread synchronization during the compute phase is unavoidable, at least try to minimize the usage.

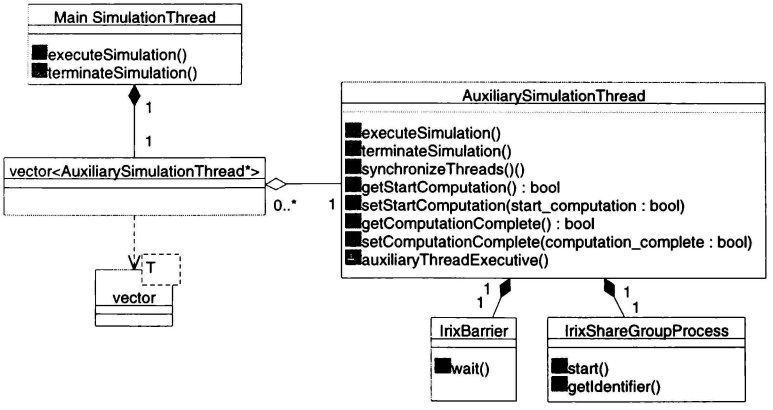


Figure 3: Computation Threads Class Diagram

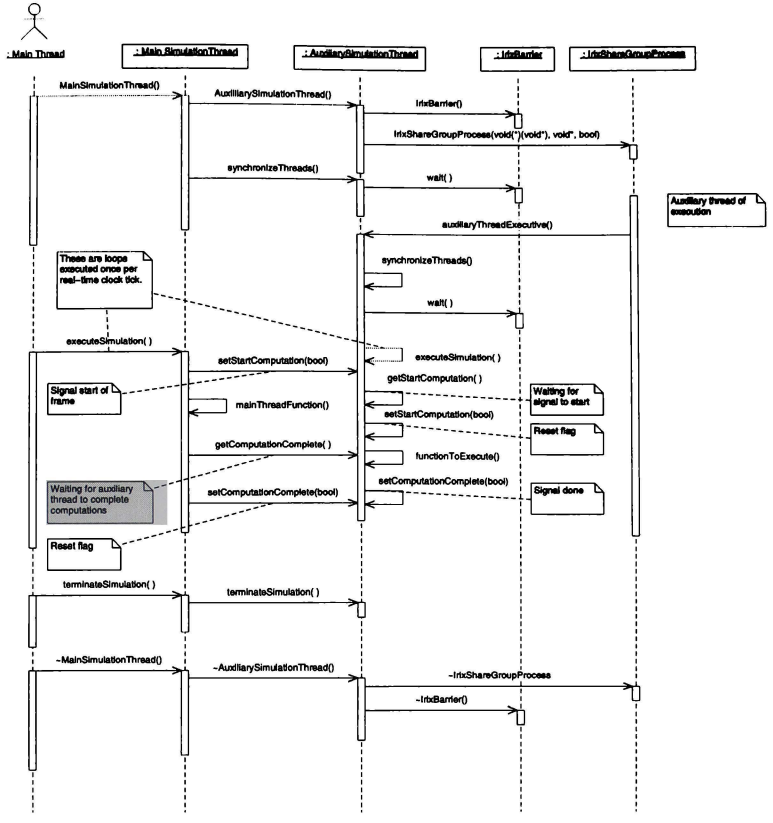


Figure 4: Computation Threads Sequence Diagram

Separate Thread for Graphical User Interface

A separate thread can be used to execute a GUI capable of monitoring and controlling a simulation. The GUI is usually considered non-real-time, so the GUI thread is assigned to run on a non-real-time processor. Due to the fact that many GUI toolkits are not thread-safe, the GUI is designed such that only a single thread actually executes function calls to the GUI toolkit.

LaSRS++ provides an interface class that simplifies the creation of separate thread GUIs. `class`

`SeparateThreadGui` is an abstract class that encapsulates GUI thread creation and synchronization. This class specifies the behavior that all separate thread GUIs must exhibit. All GUI startup synchronization is also handled within the interface class constructor. Since this class specifies the execution sequence, developers can focus on GUI design and implementation issues, without being concerned with thread-specific issues.

A new GUI is created by deriving a new class from `SeparateThreadGui` and providing definitions for the following pure virtual functions:

- `installSignalHandlers()`
- `createGuiObjects()`
- `executeEventLoop()`
- `destroyGuiObjects()`

These member functions contain GUI toolkit function calls that create, execute and destroy the actual GUI. The actual implementation can be performed without any knowledge of threads. This design allows the GUI developer to focus on GUI design and implementation issues, not thread issues.

`SeparateThreadGui` class contains all the necessary thread and synchronization objects (see Figure 5). The constructor creates a separate thread that executes private member function `guiThreadExecutive()`; this is the predefined execution sequence. In this design, the main thread creates an instance of a concrete class derived from `SeparateThreadGui` (see Figure 6). Startup synchronization with the new thread usually occurs in the constructor of the new derived object.

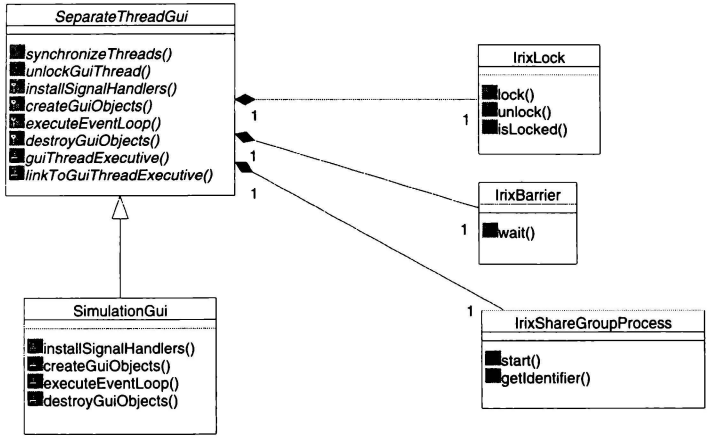


Figure 5: Separate Thread GUI Class Diagram

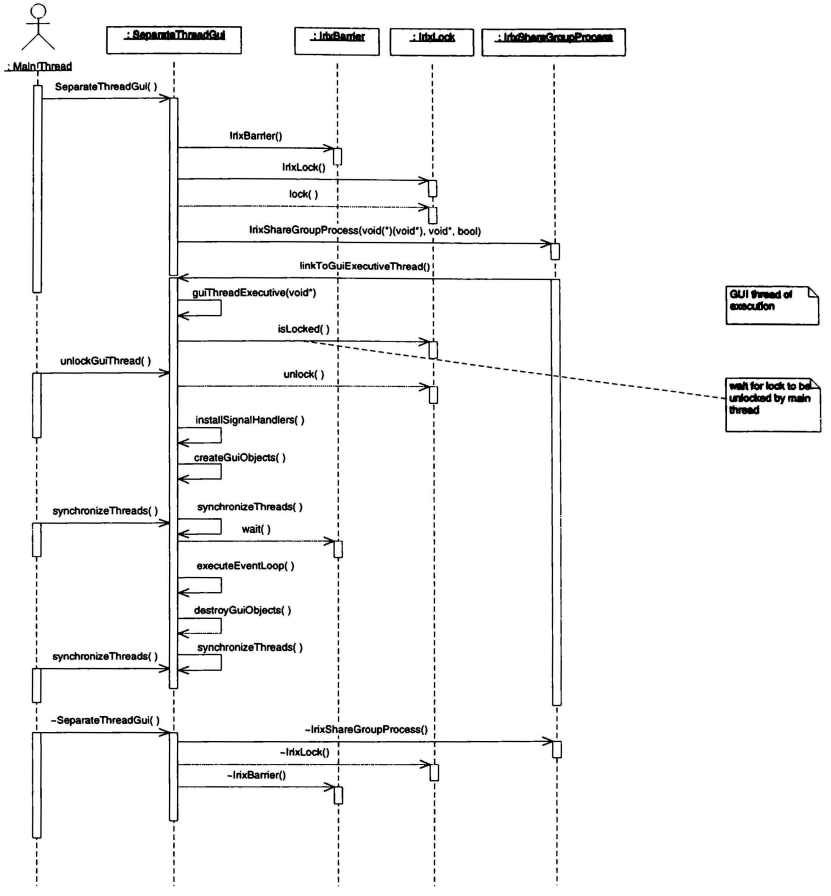


Figure 6: Separate Thread GUI Sequence Diagram

Developers and users want to monitor and modify simulation variables. Monitoring simulation variables does not require any thread synchronization to avoid data corruption. There are several techniques to minimize synchronization between a separate thread GUI and the other real-time threads when modifying simulation variables [2]. One technique is to allow the GUI thread to only change variables that are read-only to the simulation threads. Another method involves having the GUI modify values in a temporary buffer. The real-time threads can then use the updated buffer values to modify simulation variables when there is no risk of data corruption.

Conclusions

The NASA Langley Standard Real-Time Simulation Framework in C++ (LaSRS++) provides support for multithreaded programming. The main motivation for using multiple threads is to make parallel, real-time simulation programming easier for the developer. The presented object-oriented designs provide an environment where developers can focus on their specific programming task instead of getting bogged down in thread details. Multithreaded programs are developed as single entities instead of several cooperating programs. Two of the possible uses of multiple threads include: increasing the number of computations possible during a frame and executing graphical user interfaces for program control and data monitoring.

Bibliography

[1] David R. Butenhof. *Programming with POSIX Threads*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1997.

[2] Michael Madden, et al. Constructing a multiple-vehicle, multiple-cpu using object-oriented C++. Number AIAA-98-4530 in AIAA Modeling & Simulation Technologies Conference, August 1998.

[3] Richard A. Leslie, et al. LaSRS++ an object-oriented framework for real-time simulation of aircraft. Number AIAA-98-4529 in AIAA Modeling & Simulation Technologies Conference, August 1998.

[4] Robert Martin. *Designing Object-Oriented C++ Applications Using the Booch Method*. Prentice-Hall, Inc., 1995. ISBN 0-13-203837-4.

[5] Scott Meyers. *Effective C++*. Addison-Wesley Publishing Company, 1992. ISBN 0-201-92488-9.

[6] Pierre-Allain Muller. *Instant UML*. Wrox Press Ltd., 1997. ISBN 1-861000-87-1.

[7] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Publishing Company, third edition, 1997. ISBN 0-201-88954-4.

[8] Topics in irix programming. Technical Report 007-2478-006, Silicon Graphics Incorporated, 1998.

[9] Uresh Vahalia. *UNIX Internals*. Prentice Hall, Upper Saddle River, New Jersey, 1996.

A GENERIC OBJECT-ORIENTED IMPLEMENTATION FOR FLIGHT CONTROL SYSTEMS

Patricia C. Glaab, Michael M. Madden*

Unisys Corporation
NASA Langley Research Center
Mail Stop 125B
Hampton, VA 23681

Abstract

This paper presents a design for a generic flight control system (FCS) architecture that breaks the control system into a coupled interaction of laws and devices with a standardized method for execution. Laws generally are computational components that generate commands as outputs, and any number of laws can be isolated and registered on a list in any order for execution. Control devices are code components that receive the command inputs and use additional computations to generate device outputs, such as the servoactuator positions for the control of surfaces. Any number of devices are allowable for a given flight control system and are registered in list format for execution. This method allows for both simplistic FCS implementations and highly complex control systems without changing the architectural requirements of the high level executive.

By separating the laws from the devices, special case handling required for control law bypassing (such as that required for direct-drive surface testing and linear analysis) is easily handled at the execution level. No special code support is required internal to the laws.

Introduction

This paper presents a method for structuring flight control system code as part of an object-oriented simulation framework. The design presented was developed at NASA Langley Research Center by Unisys Corporation for use within the Langley Standard Real-Time Simulation in C++ (LaSRS++). The goals for the original design included several objectives. The first objective was to develop a common software heirarchy that could effectively handle FCS requirements for different styles of aircraft. The parent classes for the elements would reside in the simulation framework. Ideally, this structure would simplify installation of new FCS models being added while providing a common execution style that only required one documentation effort.

A second objective was to allow special FCS processing as part of the common executive. Specifically, direct manipulation of surface positions was required for linear analysis and open-loop checkcase matching. This design was developed to allow this type of operation without embedding special handling instructions in the aircraft-specific FCS code.

To achieve the solution, an architecture was developed that reduces a given FCS to a collection of *laws* and *devices* and treats them as separate components. Once this is done, the common elements in seemingly unique control systems becomes apparent. The flexibility of the system is maintained through the use of variable length lists to control the laws and devices. This method effectively accommodates aircraft with varying ranges of complexity and sizes of models.

Design Objectives

Aircraft that differ in both form and function necessarily have unique technical requirements for their flight controls. The differences between a military fighter aircraft FCS and that of a commercial

Copyright © 1999 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

transport at first inspection may seem more profound than the commonality. Yet, in order to satisfy the design goal, a methodology that serviced each style effectively was required.

In this paper, the implementations of an F16A and a Boeing 757 aircraft are used to demonstrate the application of the final design to two very different aircraft models. The F16A simulation model uses a low-speed subset of that aircraft's control laws. The FCS outputs one signal for aileron, stabilator, leading edge flap, speedbrake, and rudder as required to interface to the supplied aerodynamics model. The 757 simulation model uses a full-envelope FCS model and supplies surface deflections for left and right ailerons, elevators, flaps, and slats, a single rudder and stabilizer, and twelve spoiler deflections as required for its aerodynamics model. The 757 uses a complex servoactuator model with hinge moments, hydraulic system options, and variable rate and position limits. The F16A uses a simple first-order servo calculation with fixed rate and position limiting. The F16A computes its commands to surfaces using only a longitudinal, lateral, and directional control law. The 757 also uses laws for the three axes, and additionally uses a high lift, a spoiler, and a yaw damper control law. Yet both are very effectively modeled using the architecture described here.

The approach used in the design of this FCS architecture was to ignore the specific type of surfaces (or devices) and the intricacies of generating the specific commands to drive them, and to reduce the level of complexity to a group of *laws* that act upon a group of *devices*. In this way, the software is modeled similar to the actual aircraft. In an actual aircraft, some combination of pilot inputs and synthesized signals from a flight control computer are fed to the control surface servos. The actual deflection that results from the command may depend on the current aircraft states or environmental conditions, depending on the complexity of the simulation model. The derivation of commands and the surface movements can be treated as separate entities.

Once this separation of laws and devices is made, the similarities between very different styles of FCS become quite apparent.

Laws

A control law is the set of computations that provide a command as an output. The internal computations may represent flight control computer modeling, mechanical linkages, etc. Inputs to control laws are generally pilot inputs from the cockpit and aircraft state variables. The outputs are the commands to the devices.

Control laws may be loosely or tightly coupled, depending on the aircraft model. In a very simplistic configuration (such as a general aviation aircraft without any autopilot), the laws may be simply the interaction of mechanical connections that transform pilot command inputs into commanded force or deflection outputs. In a more complex model, a stability augmentation system or a flight control computer may generate or modify commands.

This FCS class architecture design cares neither about the number of laws used nor the internal complexity. The parent *ControlLaw* class, which resides in the general framework, defines methods required by client aircraft-specific control law classes which inherit from it. By maintaining a variable length list of laws, ControlSystem simply operates upon each law registered to it at the referenced location. In the C++ implementation, the aircraft-specific FCS instantiates the laws required and registers a pointer for each law during construction.

ControlSystem contains a pure virtual function called *execute()* which must be specifically defined by each client control law. Making the method pure virtual insures definition of the method within each client. During FCS processing, the control system executive steps through its list of registered laws and simply calls the *execute()* method for each. Varying number of control laws are accommodated for different aircraft models with the same high level controller code.

Devices

A device is a software representation of what would be hardware on an actual airplane. Surfaces are the primary type of device used, but a device can be any aircraft component that augments the flight. Side-thrusters would be another device example. Inputs to devices are usually a command signal and state variables or environmental variables that effect the device's movement. Most modern devices use some type of servoactuators to achieve output deflections.

The flexibility for number and complexity of devices is maintained similarly by a variable length list maintained by the parent *ControlSystem* class. In the C++ implementation, the aircraft-specific FCS instantiates the number and type of devices it requires for its model and registers a pointer for each to the

device list at construction time. The *ControlDevice* parent class, from which all control devices inherit, contains a pure virtual method called *drive()* which must be defined by each client device class. During execution, the *ControlSystem* executive steps through its list of devices and calls the *drive()* method for each.

Modeling of Common Behaviors

The computational behaviors common to FCS code structure and execution are also contained within the parent *ControlSystem* class. Every FCS must provide inputs to its laws, execute the laws, provide inputs to its devices, and drive them. In the standard execution, these are done by the *ControlSystem* class. The *ControlSystem*, from which the aircraft-specific FCS inherits, defines the pure virtual methods *directInputsToControlLaws()* and *directInputsToControlDevices()*. These must be redefined by each airplane to fit the its own unique requirements.

Additionally, the methods *evaluateControlLaws()* and *driveControlDevices()* are defined within the parent *ControlSystem* class and contain the code to step through the list of laws and devices. These need not be refined at the aircraft-specific level. They are, however, virtual methods that may be redefined if some unusual execution is required.

One other method is provided as virtual by the *ControlSystem* class for use at the client's discretion. The *compositeDeviceCalculations()* method is provided to compute combinations of device outputs required for other systems in the simulation. For example, in the 757 aircraft, this method is used to compute *slat_average* and *elevator_differential* which are exported to the aerodynamics model and the data recording model.

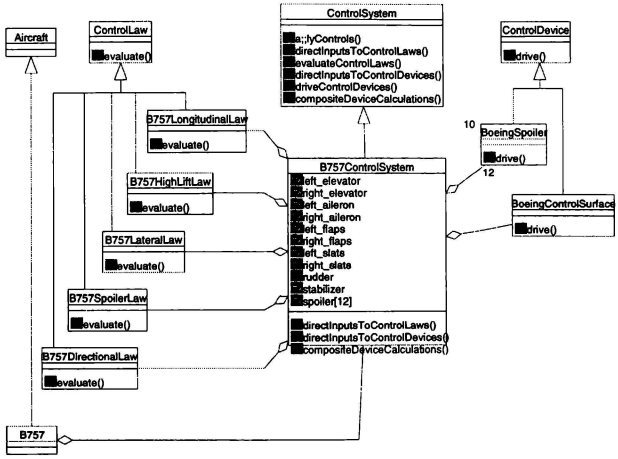


Figure 1- 757 FCS Class Structure

Execution Options

Since this methodology was developed to support special processing requirements for linear analysis and open loop checkcase matching, alternate processing flexibility was built into the architecture at the parent class level. A flag is provided by the *ControlSystem* class called *open_loop*. When the *open_loop* flag is set true, an alternate method of execution is followed. During open-loop processing, execution is defined in its entirety by the client aircraft control system via a virtual method called

useTrimCommandsAsResponse(). Within the client's version of this method, commands or deflections can be directly assigned as necessary for special handling. The executive section is simply defined in *useTrimCommandsAsResponse()*, and the *open_loop* flag set true.

Inheritance Structure

Figures 1 and 2 show the inheritance structure for the 757 commercial transport and the F16A military fighter within the generic FCS architecture. The top-most boxes on each diagram represent the framework classes: *ControlSystem*, *ControlDevice*, *ControlLaw* and *Aircraft*. Here the *B757* executive class, which inherits from *Aircraft*, has 21 control surfaces and 5 control laws to generate its model. The *F16A*, by comparison, only uses three control laws and five control surfaces to define its FCS model. The pure virtual methods, *evaluate()* and *drive()*, initially defined in the *ControlLaw* and *ControlDevice* classes respectively, are redefined in the aircraft-specific classes which inherit from them. Pure virtual classes *directInputsToControlDevices()* and *directInputsToControlLaws()*, initially defined in *ControlSystem*, are redefined for the aircraft-specific control system classes. Since each aircraft requires combined surface outputs, each redefines the *compositeDeviceCalculations()* method. Neither, however, defines *evaluateControlLaws()* or *driveControlDevices()* which are handled entirely by the parent class, *ControlSystem*.

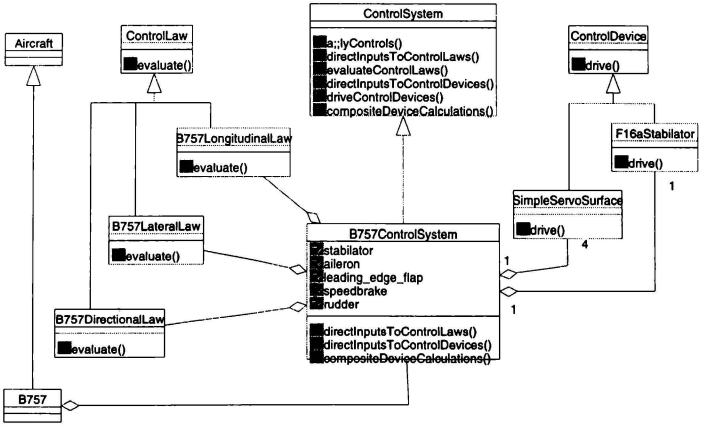


Figure 2 - F16a FCS Class Structure

Object Interaction

Figure 3 and 4 show the interaction of the objects during closed-loop operation for the Boeing 757 and F16A aircraft simulations. At construction time, the client control system instantiates the number and type of laws and devices it requires and registers a pointer to each to the list of laws contained in *ControlSystem*. The order of registration dictates the order of execution used later for each law and device by the *ControlSystem* class. The client aircraft, as part of its execution process, tells *ControlSystem* to *applyControls()*. The *applyControls()* method within *ControlSystem* calls *directInputsToControlLaws()*, a pure virtual method defined at the client control system level. The *directInputsToControlLaws()* method in the client control system sets inputs required for each control law that it uses. This input processing usually involves gathering state variables from other parts of the client aircraft and sending them to the passive control law objects.

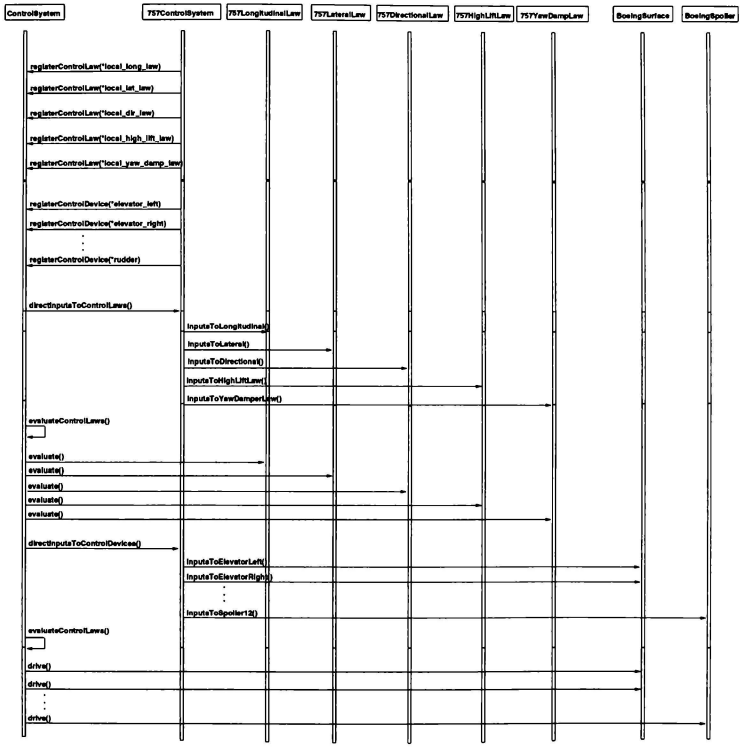


Figure 3 - 757 Object Interaction Diagram

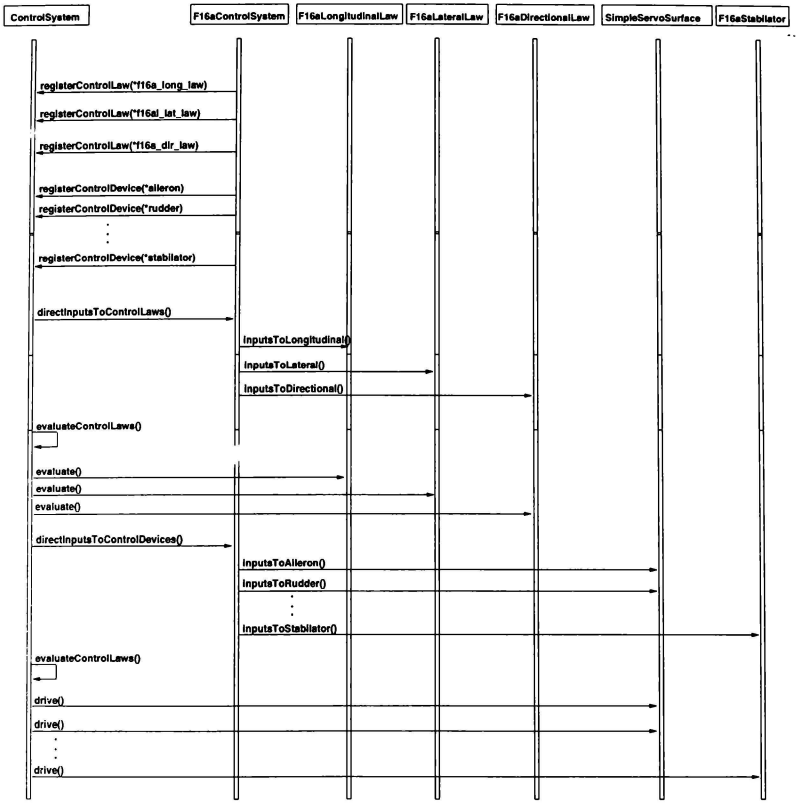


Figure 4 - F16A Object Interaction Diagram

The *applyControls()* method then calls *evaluateControlLaws()* which simply steps through its list of control laws and calls evaluate() for each. The *evaluate()* method is defined in the parent *ControlLaw* class as a pure virtual function, and must be redefined by each specific control law that inherits from it. All technical code required to compute the command output is contained in the *evaluate()* method within the clients control laws or in local scope methods called by it.

Similar processing is then performed by the *ControlSystem* class for device calculations. The *directInputsToControlDevices()* method is defined at the client control system level and sets inputs required for each control device used. This input processing usually involves gathering commands from the control laws and additional aircraft state variables from other simulation objects and updating the input data within each device object.

The *driveControlDevices()* method in *ControlSystem* is called which steps through the list of registered devices and instructs the *drive()* method to be executed for each one.

Finally, the *compositeDeviceCalculation()* method is called to compute required output combinations for other parts of the simulation.

Note that even though the number and type of laws and surfaces differ for the two aircraft, the execution methodology and the code structure for the classes remains the same.

Conclusions

The initial design and testing phase for this flight control system architecture (as is usual for sophisticated and reliable object-oriented development), required a substantial effort on the part of the developer. This method was first used for the 757 simulation model. Before this architecture was installed, the 757 FCS code was encumbered with switch statements and branch "if" testing because of the extensive amount of special processing required for that particular project. The code was very fragile and had become almost indecipherable. After the 757 control system was revamped to fit this generic method, the code was highly robust and instantly comprehensible by anyone willing to familiarize themselves with the architecture.

Since its inception, three other aircraft have used the architecture with extensive time saving both in testing and validation of the code and in the design review documentation requirements. Testing and coding for these aircraft was practically limited to the installation of their technical equations and inputs to them. Linear analysis capability was effectively free. The development effort was easily justified in this instance, if not absolutely required, for the 757 model to be maintainable through its projected life span. Smaller control systems that capitalized on the effort would probably not have expended the initial development effort unless a large amount of special processing was expected in that project's life.

A common method of class structure and execution for a framework, however, is a large motivator. Installation and testing of control systems, previously handled exclusively by experienced simulation developers, is now effectively accomplished by relatively new developers in the LaSRS++ framework. The conclusion is that the initial testing and development effort is highly recommended for several situations:

when the complexity of one particular model compromises the code's readabilty
when many aircraft share a common framework
when the responsibility for code maintenance will be handled by changing personnel over the project's life span

Bibliography

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Massachusetts, 1995.

[2] Steve McConnell. Code Complete: A Practical Handbook of Software Construction. Microsoft Press, Redmond, Washington, 1993.

[3] Terry Quatrani. Visual Modeling With Rational Rose and UML. Addison-Wesley, Reading, MA, 1998.

[4] Scott Meyers. Effective C++. Addison-Wesley, Reading, Massachusetts, second edition, 1998.

[5] Grady Booch. Object-Oriented Analysis and Design. Benjamin/Cummings, Redwood City, California, 1994.

[6] R. Leslie, D. Geyer, K. Cunningham, M. Madden, P. Kenney, P. Glaab. LaSRS++: An Object-Oriented Framework for Real-Time Simulation of Aircraft. AIAA-98-4529, Modeling and Simulation Technology Conference, Boston, MA, August 1998.

AIAA-99-4340

USING ABSTRACTION TO CREATE A PORTABLE OBJECT-ORIENTED SIMULATION

P. Sean Kenney and David W. Geyer

Unisys Corporation
NASA Langley Research Center
Mail Stop 169
Hampton, VA 23681

Abstract

A common problem faced in the design of an object-oriented simulation is that there are many different groups of end users of the simulation framework. Each group invariably has a different computer platform on which to run the simulation. In an attempt to satisfy all of these different groups, a simulation framework should be designed to be portable so that it can be operated on as many different platforms as possible. The purpose of this paper is to describe several designs that isolate the framework from the platform dependent services required by a simulation.

Introduction

Like most complex applications, a flight simulation framework requires a large amount of operating system support to perform its tasks. POSIX, the Portable Operating System Interface, is a standard intended to allow an application to move from one operating system to another by simply recompiling it [1]. POSIX is an evolving, growing standard that provides the same interface to system calls on operating systems that support the standard. There are three problems involved with relying on POSIX to provide platform independence:

1) Many platforms do not have POSIX support.
2) Some platforms claim to have POSIX compliance when they only support a minimal POSIX implementation.
3) For a given platform, there may be platform specific services that are not provided in POSIX that could be taken advantage of by a simulation framework.

Until POSIX support is universal and completely supported by most operating systems, another solution to create portable a simulation framework is required.

To provide portability, an object-oriented simulation framework must address three issues. First, the framework must encapsulate the behaviors of operating systems and real-time environments by using abstraction [2] to define interfaces for platform dependent functions. Next, the framework cannot be coupled to any simulation hardware. The framework should isolate complex simulation models from the simulator hardware interfaces. By de-coupling the framework from the hardware, the framework can run on platforms connected to different hardware devices or no hardware devices at all. Finally, if a framework uses a Graphical User Interface (GUI) as a means to control the simulation during execution, the framework cannot be coupled to the GUI. This provides the greatest flexibility for the system as a whole.

The remainder of this paper will discuss each of these issues and how they were dealt with in the Langley

Copyright © 1999 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

Standard Real-Time Simulation in C++ (LaSRS++) Application Framework. LaSRS++ provides a powerful object-oriented framework for dynamic vehicle simulation in real-time [3]. The framework's object-oriented design makes the software extremely flexible, easily maintainable, and provides a high degree of reuse. The LaSRS++ framework currently supports hard real-time simulation on the SGI Onyx and the Convex C3800 platforms. The framework has also been run in a soft real-time mode on SGIs running Irix, Sun workstations running SunOS and Solaris, IBM RS6000s running AIX, and IBM PCs running Linux and Microsoft NT.

The abstractions presented in this paper are the product of an iterative object-oriented design process. The designs provide decoupled, unit-testable, and complete interfaces to platform specific resources. The abstractions are intended to simplify the complexity of porting a simulation framework to new platforms.

Platform Dependent Services

A large number of operating system services are used by a simulation framework to perform real-time simulation. A framework must address scheduling, timing, data sharing, synchronization, I/O, and many other problems. Operating system services provide solutions to these problems but directly tie the framework to a platform. A framework must use a design that isolates operating system implementation details from the framework. Such a design allows the framework to use timers, schedulers, shared memory, semaphores, and other operating system resources in a portable fashion. LaSRS++ uses an elegant design employing the Abstract Factory, Bridge, and Singleton design patterns to completely isolate the framework from the implementation details required to use operating system services [4].

The design will be described using two common operating system services, shared memory and semaphores. Both of these services are commonly used in simulation frameworks, and the required system calls differ greatly from platform to platform.

Shared Memory

Most modern operating systems provide a means to map memory into the memory spaces of two or more processes at once thereby allowing the processes to share data. This mapping is known as shared memory. The system calls used to create and access a shared memory segment vary from platform to platform. Using object-oriented techniques, it is possible to present users on all platforms with a common interface to shared memory but allow the actual low-level implementation to vary according to the platform being used. This is accomplished by using several different object-oriented design patterns. Design patterns describe simple and elegant solutions to specific problems in object-oriented software design.

Figure 1 uses the Unified Modeling Language (UML) to show the class diagram for the shared memory interface software.

Bridge Pattern

The Bridge pattern decouples an abstraction from its implementation.[2] This design uses the Bridge pattern to isolate client code from the platform specific details of a specific implementation. The approach is to have clients use an abstraction object that forwards its public member function class to a hidden platform specific implementation object. The abstraction object uses the implementation object through the pure polymorphic interface defined by the abstract implementation base class. In this case, a shared memory interface object (SharedMemory) interacts with a platform specific shared memory implementation object (SharedMemoryImpl) through a polymorphic interface. An appropriate concrete implementation class is defined for each platform on which the simulation framework is executed upon. The appropriate concrete implementation object for a given platform is selected at run-time.

Abstract Factory Pattern

The Abstract Factory pattern is used to create the correct instance of the platform specific shared memory implementation object at run time. This creational pattern provides an interface for creating families of related objects without specifying their concrete classes. In this case, the family of related objects are

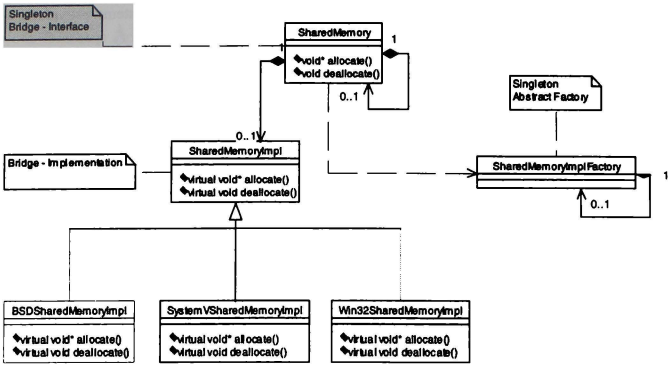


Figure 1 – Shared Memory Interface

the platform specific implementation objects. The abstract factory object (SharedMemoryImplFactory) contains knowledge of the specific platform that is being used. The constructor for the shared memory interface object invokes the makeSharedMemoryImpl member function of the abstract factory object. This member function uses knowledge of the specific platform to return the appropriate implementation object for the given platform. This specific implementation object is stored as a hidden attribute of the shared memory interface object.

Singleton Pattern

The Singleton creational pattern is used whenever it is necessary to ensure a class only has one instance, and provide a global point of access to the single instance. The shared memory interface class and the abstract factory class are implemented as singletons to provide a global point of access for creating and accessing a shared memory space.

Design Advantages

This design has several advantages that address the issues of portability and maintainability:

1. *Client code is de-coupled from the platform specific shared memory implementation details.* This de-coupling allows changes in the shared memory implementation classes to have no impact on the client code; i.e., the client code does not need to be recompiled if the implementation changes. By coupling client code only to a common interface, the client code becomes platform independent. This allows client code to be portable to any platform supported by the shared memory interface class.
2. *Only a single class needs to be written to support a new platform.* Adding support for a new platform involves two steps:
   (a) Write a new shared memory implementation class that interfaces with the platform specific shared memory routines
   (b) Add to the Shared memory implementation abstract factory the capability to create the new shared memory implementation object

3. *Creation of platform specific objects can be isolated to a single abstract factory object.* The shared memory interface class only references the abstract shared memory base class. All concrete implementation details are confined to the implementation abstract factory.

4. *The design is generic and can be applied to other operating system services.* The following section describes how the design was applied to another important operating system service - semaphores.

It is important to note that the concrete implementation classes cannot usually be compiled on any platform other than the one the class is intended. Makefile directives are the most common method of dealing with this issue.

Semaphores

A semaphore is a synchronization object that maintains a count between zero and a specified maximum value. The count is decremented every time a thread or process "acquires" the semaphore object and incremented every time a thread or process "releases" the semaphore. When the count reaches zero, no more threads or processes can successfully acquire the semaphore and will not proceed until it can. Semaphores are useful in controlling shared resources and synchronizing multiple threads or processes.

Figure 2 shows the class diagram for the semaphore interface software as implemented in LaSRS++. The semaphore interface software design uses the same patterns as found in the shared memory interface software. The only difference is that the System V implementation class is an abstract class rather than a concrete class. Two new concrete classes derive from the SystemVSemaphoreImpl class. These concrete classes provide the implementation details for these two platforms that are different among the two System V Unix platforms.

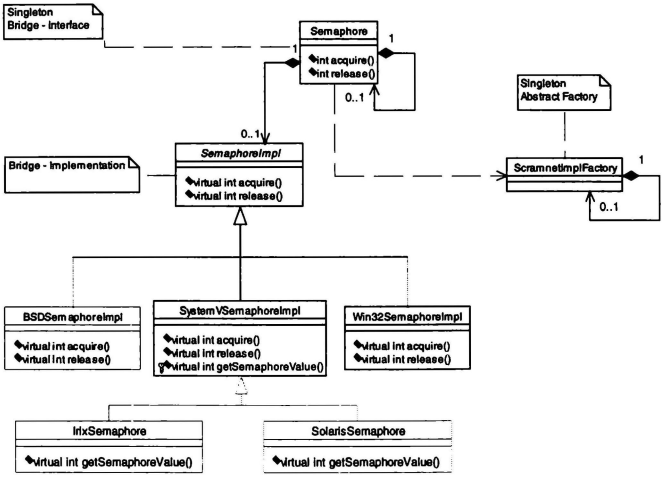


Figure 2 – The Semaphore Interface

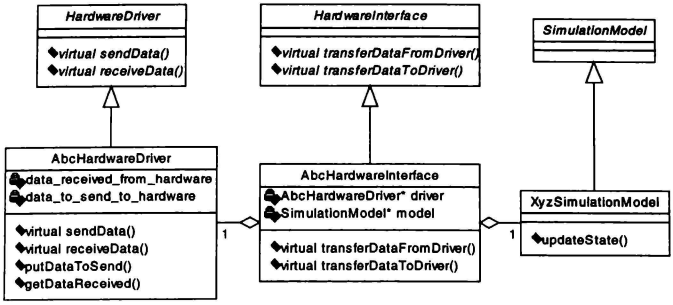


Figure 3 - Drivers, Interfaces, and Simulation Models

As described above, the design is very generic. LaSRS++ employs the design to handle several other operating system services.

The Hardware Abstraction

The hardware abstraction is composed of three main components: drivers, interfaces, and builders [5]. The drivers are the classes that actually transmit and receive data with the simulator hardware devices, the interfaces are communication classes that pass data between a simulation model and a driver, and the builders construct all of the appropriate drivers and interfaces as needed. Figure 3 demonstrates the relationships between the drivers, the interfaces, and the simulation models.

Mediator Pattern

The Mediator design pattern keeps classes from referring to each other explicitly and encapsulates how the set of classes interact.[2] The strongest asset of the Mediator design pattern is that it completely decouples the two classes from each other. It should be used whenever two classes are unrelated but need to communicate with each other.

Drivers

The driver classes are the classes that actually transmit and receive data with the simulator hardware devices. They typically contain buffers to hold the data that is transferred with the hardware and member functions to access or modify the data buffers. In the above diagram, the class AbcHardwareDriver defines the two virtual functions declared in the abstract class HardwareDriver. These virtual methods send and receive data. The class also defines methods that access and modify data transferred to and from the hardware. The driver class therefore provides the abstract interface of HardwareDriver and an interface specific to the "Abc" hardware. The driver classes are an implementation of the Bridge design pattern.

Interfaces

Interfaces are communication classes that pass data between the driver and the simulation models. The interface class also performs any manipulation of the data before transferring the data to its destination. The interface class is essentially a one way or two way data pump between the driver and the simulation model. In the illustration above, the AbcHardwareInterface is given a reference† to the AbcHardwareDriver and the XyzSimulationModel when it is instantiated. The class would then use the two references to transfer data between the two classes when appropriate. The interfaces are a variation of the Mediator design pattern.

† Unless stated otherwise, reference refers to both the reference and pointer types in C++.

Builders

Builder classes construct all of the appropriate drivers as requested by the user and construct all of the corresponding interfaces required by the simulation models. The builder classes provide an interface for creating the driver and interface classes without other simulation classes having knowledge of the particular concrete classes. The builder classes are implemented using the Abstract Factory design pattern.

Design Advantages

The advantages of this design are:

1. *The simulation models are completely decoupled from the hardware driver classes.* This allows the models to be tested with or without simulator hardware. The behavior of the simulation models with different inputs can be fully tested offline allowing comprehensive analysis of performance without using valuable simulator hardware resources. Once the performance of a model has been validated, the hardware inputs/outputs can be used to validate the model's performance in the simulation. This minimizes the validation required of new models. The decoupled simulation models also remain portable. A polymorphic class hierarchy allows different computation models to be incorporated into the simulation and use the existing hardware interface class without modification. The model classes may be exported to other sites without requiring any modifications for use. A model imported from another site can be "wrapped" in a class that has the interface required by the existing hardware interface, thereby quickly assimilating the new computational model into the simulation.

2. *Modifications to simulator hardware only require a change to the driver class.* Many hardware devices receive major and minor modifications over their lifetimes. Minor modifications are often changes to software, buffer sizes, etc. and require little or no change to the software used to communicate with the device. Major modifications may require a significant change to the software used to communicate with the device however. Because the driver encapsulates all of the code involved with direct hardware communication, the simulation is completely isolated from the modifications.

3. *The hardware driver classes can be unit tested.* Any modifications to a driver class can be tested without the simulation model. A diagnostic program can be written that uses the driver to communicate with the hardware. The diagnostic program serves two functions. It can be used to test any changes made to the driver program and it can be used to verify the operation of the hardware prior to use by the simulation. Software configuration management eases the burden of testing the new driver class by allowing the user to verify that the hardware is operating correctly with a previous version of the driver before testing the new version. (This is usually not possible when the hardware has been modified).

4. *The driver and/or interface class may be used to emulate the hardware.* Often a simulation uses real-world hardware like a flight management computer to assist in research or testing. A software emulation of a hardware device can be placed in either the driver or interface class to allow the simulation to perform necessary communications with the emulated hardware when the real hardware is unavailable. The hardware emulation may also be modified to conduct research experiments.

5. *Changes to the models can not affect the communication between a driver and it's respective hardware.* A modification to a simulation model may result in bad data being transmitted to a hardware device, but it can not cause a connection loss or crash if the hardware interface class properly limits the data being sent to the hardware device.

6. *The hardware interface classes are generally very trivial.* The classes simply use the accessor and modifier methods of the driver class and the simulation model to transmit data. Any calculations required when manipulating the data can easily be verified through testing.

7. *The hardware interface classes can often be reused by different simulation models without modification.* If the models share a common base class and the hardware interface only uses a reference to the base simulation model then no modification is required for use with different simulation models.

8. *Hardware driver and interface classes are only used on platforms that support their use.* If the framework is run on a platform that has no support for a particular hardware device, then the abstract factory builder class will not attempt to create these classes. The builder classes are the only class dependent upon hardware driver and interface classes. Because only the builder classes are dependent on the hardware classes, the framework is both maintainable and portable.

9. *Only two classes need to be written to support a new hardware device.* Adding support for a new hardware device involves three steps:

   (a) Write a new hardware driver class that communicates with the new hardware device

   (b) Write a new hardware interface class that transfers data between the driver and the simulation model as needed

   (c) Add the new classes to the abstract factory builder classes to create the new driver and interface as needed.

GUI Isolation

Simulation frameworks often use GUIs to control and/or monitor simulation states during execution. To ensure portability and maintainability, a framework cannot be coupled to the GUI. This provides the greatest flexibility for the system as a whole.

Two designs are available that allow a GUI to control a framework with minimal coupling. The first design involves using a shared memory segment as a means to allow the GUI access to simulation states [6]. This method requires the framework to instantiate all objects that are manipulated by the GUI into shared memory. The GUI can then attach to the shared memory segment and manipulate objects using their public methods. The design has the drawback that it requires framework classes to be instantiated into shared memory.

Another design involves creating a second thread to create and manage a GUI [7]. In this design a framework class must create and start the new thread. The GUI is then able to monitor and/or control simulation states because it shares the same memory as the simulation framework. Due to the fact that many GUI toolkits are not thread-safe, the GUI must be designed to only allow a single thread to execute function calls to the GUI toolkit.

It should be noted that in both of these designs, the framework is de-coupled from the GUI while the GUI is directly coupled to the framework. All communication between the framework is unidirectional. The GUI monitors and controls simulation states. The framework is completely unaware of the presence of the GUI.

Determining which GUI library or toolkit to use is another difficult issue. Three options are available to a framework designer:

1) Use the libraries that came with a platform/operating system/compiler.

2) Use commercial or free libraries from other sources.

3) Develop a library suite for each of the platforms needed in house.

Each option has its advantages and disadvantages. Using the libraries commonly found on a platform is an inexpensive solution but can limit portability. For example, most UNIX platforms have native support for X windows and ship with other graphical libraries like Xt and Motif. If Motif is selected as the GUI library for a framework, then the framework is limited only to UNIX platforms that also support Motif.

Using commercial libraries or free libraries is an excellent alternative to using the libraries shipped with a platform. Problems may arise however in finding a product that supports all off the target platforms. Support problems may also impact the development of framework features. If errors are found in these prod-

ucts, development of new framework features or framework corrections may not be possible until after the vendor addresses the problem. Another problem arises when the platform requires on operating system upgrade and an operating system compatible library is not available.

Developing a library in house provides a framework staff with the greatest flexibility but has tremendous resource requirements until the libraries are full implemented.

The GUI toolkit used by LaSRS++ has undergone several transitions since the framework was first created. Originally the framework was only run only on UNIX machines. The Motif library was used to develop a GUI to control the simulation while it was executing. The shared memory GUI design was used to allow the GUI to monitor and control the framework. Because Motif is not object-oriented, the GUI code base quickly became extremely large and awkward to maintain. The Motif library was abandoned in favor of an object-oriented toolkit called *Amulet*. The Amulet base GUI also used the shared memory design. Amulet is a GUI toolkit developed at Carnegie-Mellon University and supports almost every flavor of UNIX, Microsoft Windows, and even the Macintosh. The library was also freely downloadable over the Internet. Unfortunately DARPA funding of the Amulet project ended, and the product is no longer being developed or supported by its creators. The loss of Amulet support was a mixed blessing – a new toolkit was selected that required the GUI to be rewritten yet again. The new toolkit provides a better object-oriented foundation than Amulet and has better performance. Currently, LaSRS++ uses the free GUI toolkit called *gtk--*, a GNU software package. The library is also object-oriented and supports most UNIX platforms, Microsoft Windows, and the Macintosh. The LaSRS++ framework was also modified to use the thread based GUI design. Because of the framework is de-coupled from the GUI, users have the option to run with either the Amulet based GUI or the gtk-- based GUI. This demonstrates the importance of framework/GUI isolation. While the gtk-- based GUI was being developed, other framework users where continued to use the older GUI. This affords users the convenience of transitioning to the new GUI according to their timetable.

Conclusions

Design patterns were used to maximize code reuse while encapsulating implementation details within platform specific classes. The Bridge, Factory, and Singleton patterns were used extensively throughout the design to achieve a portable framework. The Factory pattern allows the framework to know about abstract classes and nothing about the subclasses that define the platform specific operations. The pattern establishes an interface for creating subclasses without requiring any knowledge of the subclasses. The Bridge pattern decouples an abstraction from its implementation. This allows the abstraction and implementation to vary independently. The pattern provides flexibility in that the abstraction and implementation are in separate class hierarchies. The Singleton pattern ensures only one instance of a class exists and provides a global point of access to it. Essentially, the above patterns allow the entire body of code that composes a framework to be unaware of which platform the framework is operating on.

Because the hardware interface is abstracted away in a manner that keeps the actual simulator hardware communications hidden from the rest of the framework, the framework is a robust, portable, and easy to maintain simulation system. Continual reuse of the hardware abstraction ensures that new hardware interfaces can be easily added into the framework and that these interfaces can be easily tested and debugged with or without the hardware.

Isolation of the framework from the GUI provides the greatest flexibility when moving the framework to different platforms. In selecting a GUI library for use with a simulation framework, a designer must weigh the pros and cons of the aforementioned options and select the most appropriate solution for that particular environment.

Although the designs presented in this paper were originally designed to support flight simulation at NASA Langley Research Center, the designs could be

used in any object-oriented framework to heighten reuse, portability, and maintainability.

Bibliography

[1] Bill O. Gallmeister. *POSIX.4 Programming For The Real World*. O'Reilly & Associates, Sebastpol, California, 1995.

[2] Grady Booch. *Object-Oriented Analysis and Design*. Benjamin/Cummings, Redwood City, California.1994.

[3] Richard A. Leslie, et al. *LaSRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft*. Paper Number AIAA-98-4529, August, 1998.

[4] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.

[5] P. Sean Kenney, et al. *Using Abstraction To Isolate Hardware In An Object-Oriented Real-Time Simulation*, Paper Number AIAA-98-4533, August, 1998.

[6] Michael Madden, et al. *Constructing a Multiple-Vehicle, Multiple-CPU Using Object-Oriented C++*. Paper Number AIAA-98-4530, August, 1998.

[7] David Geyer, et al. *Managing Memory Spaces In An Object-Oriented Real-Time Simulation*, Paper Number AIAA-98-4532, August, 1998.

[8] Bruce Eckel. *Thinking in C++*. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[9] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading, Massachusetts, third edition, 1997.

[10] John Lakos. *Large-Scale C++ Software Design*. Addison-Wesley, Reading, Massachusetts, 1996.

[11] Robert C. Martin. *Designing Object-Oriented C++ Applications Using The Booch Method*. Prentice-Hall, Englewood Cliffs, 1995.

[12] Scott Meyers. *Effective C++*. Addison-Wesley, Reading, Massachusetts, 1992.

[13] Scott Meyers. *More Effective C++*. Addison-Wesley, Reading, Massachusetts, 1996.

[14] David R. Musser, Atul Saini. *STL Tutorial and Reference Guide*. Addison-Wesley, Reading, Massachusetts, 1996.

[15] Terry Quatrani, *Visual Modeling With Rational Rose and UML*, Addison Wesley, Reading, Massachusetts, 1998.

[16] Pierre-Alain Muller. *Instant UML*. Wrox Press Ltd., 1997. ISBN 1-861000-87-1.

AN OBJECT-ORIENTED DESIGN FOR RAPID MODIFICATION OF FILTERS

Michael M. Madden*

Unisys Corporation
20 Research Drive
Hampton, VA 23666

Abstract

Using object-oriented languages, developers can package data and functions into classes and relate them by the services they provide. Classes that provide the same service but use different implementations are placed in inheritance relationships. The base class declares the interface for the service, and the derived classes define the implementation. Thus, client code can be written to the unified interface, but the system behavior depends on the actual class that is constructed. This feature is called abstraction [1]. Abstraction removes decisions about system configuration from the code that models operation to the code that handles construction. The operational code is easier to read and maintain. The operational code is also more efficient because it does not redundantly evaluate configuration logic every frame. Control system development and research can benefit from the application of abstraction. In simulations used for control system research and development, transfer function definitions frequently change. The change can occur in the coefficients, the order of the transfer function, or the digital algorithm. The simulation may retain several transfer function configurations for comparative analysis or to accommodate multiple research projects. Applying abstraction to the design of transfer functions keeps the logic that manages different configurations out of the operational code.

* Senior Member, AIAA

Copyright © 1999 by the author. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

Introduction

This paper examines the advantages that object-oriented design brings to the rapid modification of transfer functions, called filters within this paper. Rapid modification is the ease by which the definition of a filter can be found and changed *correctly*†. A filter's definition includes its order, its coefficients, and the digital algorithm that implements the transfer function (e.g. a Tustin z-transform). The paper also explores a related topic, the simplicity of maintaining multiple filter configurations. Maintenance of multiple configurations includes the ease with which a new configuration can be created and the ease of identifying and correctly modifying existing configurations.

Rapid modification and multiple configurations are important in simulation projects that involve control system‡ development or research. In these projects, filter definitions frequently change; and multiple filter configurations are retained for comparison. To highlight the advantages that object-oriented design brings to this environment, the object-oriented design is compared to a procedural design. The procedural design originates from the Langley Standard Real-time Simulation (LaSRS), a simulation framework written in FORTRAN. The LaSRS design is not unique; similar implementations have appeared in simulation code re-

† Correctness is an important criterion. Between designs that aid change, a design that inhibits unintentional defects is superior to one that facilitates them.

‡ This paper defines a control system as a collection of models that simulate from the pilot controls to the control effectors; e.g., stick dynamics, control laws, and actuator models.

ceived from third parties. The object-oriented design comes from the successor to LaSRS, LaSRS++, written using C++.

The Procedural Design

LaSRS implements each pair of transfer function order (e.g. first order, second order) and digital algorithm as a function. For example, the function signature for a first order filter implemented with a Tustin z-transform is:

```
FUNCTION FOTUST(U,A,B,C,D,SCR)
```

The arguments are defined as follows: U is the input to the filter. A, B, C, and D are the Laplace-domain coefficients. SCR is an array of at least six elements in length; the array holds the equivalent z-domain coefficients and the past values of the input and output. The subroutine also has a hidden dependence on the variable T (elapsed time) and H (time step); these are accessed globally through a common block. The function returns the output of the filter calculation. The procedural design has two characteristics that complicate filter modification and maintenance of multiple configurations. First, the filter function is separate from its implementation data, i.e. the scratch array. Second, the function tightly binds a filter's definition to its execution.

The developer must create and manage a "scratch" array for each filter. The size of the scratch array differs by transfer function order (since order dictates the number of internal states that must be stored). If a new release of the control system changes the order of the transfer function, the developer must remember to change the size of the scratch array in addition to the subroutine call. Otherwise, a defect will be introduced; the program will read and write past an array boundary. Alternatively, the developer could make all "scratch" arrays large enough to accommodate the highest order filter supported by the framework. However, this increases the memory requirements for the simulation unnecessarily. The scratch array being distinct from the filter function also opens the possibility that a scratch array will accidentally be used more than once, also introducing a defect; later filters will incorrectly use the internal states and temporary calculations of earlier filters. The potential defects associated with scratch arrays are very difficult to uncover.

Multiple configurations further complicate the management of scratch arrays. The developer can create a new scratch array for each filter of each configuration; this significantly increases the number of scratch arrays that are maintained. Or, the developer can create one scratch array for all configurations of a given filter; the developer must still remember to correctly size the array each time a new configuration is added.

The filter function defines and executes the filter. Thus, the filter definition is ingrained in the control system operational code. If a change is made to either the transfer function or the digital algorithm of a filter, the change appears in the operational code. Since filter executions are dispersed throughout the operational code, the definitions are also isolated; the developer cannot collect filter definitions in a central location for easy inspection and modification. In large control systems, the developer will traverse thousands of lines of code to find the execution point of the filters to be changed.

The isolation of filter definitions presents a problem for supporting multiple configurations. Multiple configurations are typically selected using conditional statements (e.g. IF-THEN). The developer can place a conditional statement at the execution point of each filter in the configuration. This effectively litters the operational code with conditional statements, reducing performance and readability. To identify the configuration, all conditional statements must be located and analyzed. Alternatively, the developer can copy and tailor the operational code for each configuration. Then, one conditional statement selectively executes the operational code tailored for a given configuration. This option significantly increases code size and creates a maintenance problem. The developer must replicate changes to the operational code across all copies. In both cases, the simulation evaluates the configuration each frame of operation even though the configuration will not change during operation. The extra computation is inefficient and is better placed in the initialization code. But, the procedural design prevents it.

Representing the transfer function coefficients with variables can transplant part of the filter configuration to the initialization code. The coefficient definitions are relocated from the operational code to the initialization

code. Any change to a filter coefficient is now made in the initialization code; the operational code remains unmodified. Configurations that differ only by the coefficients are evaluated once in the initialization code; the operational code no longer performs this evaluation. However, changing the transfer function order or the digital algorithm still requires a change to the operational code. In fact, a different function is called. At best, the procedural design allows only part of the filter definition to be configurable during initialization.

The following examples summarize the steps involved in modifying filters and creating a new configuration; they represent worst case condition: order, coefficients, and digital algorithm are changed. The order of the transfer function is increased in the example.

Changing an existing filter

1) Search the operational code for the filter.
2) Change the name of the filter function to the function appropriate for the new order and digital algorithm.
3) Create new variable names for the additional coefficients.
4) Note the name of the scratch array and new coefficient variables.
5) Search the declaration code for the scratch array and expand its size for the new filter function.
6) Search for the coefficient declarations and declare the new variables.
7) Modify initialization code for the new coefficient variables.

Creating a new configuration

1) For each filter in the configuration[§]:
   a) Search the operational code for the filter.
   b) Add a conditional statement for selecting the configuration.
   c) Add the function call for the new filter.
   d) To minimize data size, reuse as many variables from the other configurations and perform steps 3-7 in the "Changing an existing filter."

[§] These instructions loop; the total number of steps is the product of the steps for one filter and the number of filters in the configuration.

   The new scratch array must be large enough to support all configurations.

These steps show that the procedural design requires changes to multiple units of code (e.g. declaration, initialization, and operation) when a filter is changed. Depending on the development process, each of these units may require review, testing, and verification. Modified code units and their dependents must also be re-compiled; as more code units are changed, the set of files that are re-compiled grows. If multiple programmers develop the operational code, placing filter definitions in the operational code also impacts configuration management. The operational code is very volatile during development. The procedural design adds filter definitions and multiple configurations as potential changes to the operational code. Thus, the procedural design can significantly increase the potential changes that must be reconciled when the work of multiple developers is combined. The more changes that must be reconciled in a unit of code, the larger the chance that a defect will be introduced during the merge process. All these factors indicate that maintenance of the procedural design can be costly.

The Object-Oriented Design

The object-oriented design eliminates the disadvantages of the procedural design by exhibiting the opposite characteristics. In other words, the implementation data is bound to the implementation; and the filter definition is separate from its execution. This is accomplished using abstraction mechanisms of encapsulation and polymorphism [1]. Encapsulation hides the implementation details of the object; it exposes only the necessary functionality. Encapsulation frees the developer from the details of creating and managing the "scratch" array for each filter. Thus, encapsulation binds an implementation with its data. Polymorphism allows the developer to name an action for a hierarchy of classes but allows each class to implement it differently. Developers can act on objects of these classes as if they were all objects of the base class but invoke behavior specific to the actual class of the object. Polymorphism allows the definition of a filter to be completely separate from its execution. The operational code executes a filter without knowing its transfer function or digital algorithm.

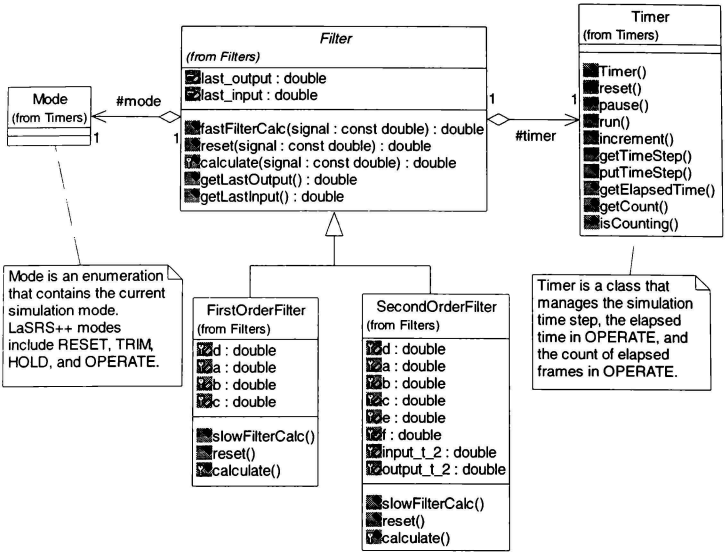


Figure 1 Filter Class Diagram

Figures 1 and 2 display the object-oriented filter design using unified modeling language (UML) notation [4]. Figure one shows a top-level view of the filter classes. Figure two illustrates how classes of digital algorithms (simple z-transforms in this case) are represented and associated with a filter. For ease of understanding, the figures show a simplification of the LaSRS++ design, focussing on filters using simple z-transforms as their digital algorithm[1].

The design employs the strategy design pattern. The strategy pattern "defines a set of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from the clients that use it [6]." The Filter class is the base class of the hierarchy. As exemplified by the fastFilterCalc() method, the Filter class defines a filter to be an object that accepts an input, performs a calculation on the input, and produces an output. fastFilterCalc() consists of a case statement that performs reset() or calculate() based on simulation mode. reset() returns either the steady state output for the given input (default) or a user-defined initial output. calculate() executes the digital algorithm. reset() and calculate() are not defined; they are polymorphic methods. The derived classes must define them. The order of the transfer function distinguishes the derived classes. The order determines the number of Laplace-space coefficients that defines the filter (i.e. a, b, c, d, e, and f) and the amount of implementation data required. Thus, the order is the next level of commonality between filters.

Unlike the procedural functions, the filter classes are not subdivided by digital algorithm. The FirstOrderFilter and SecondOrderFilter classes can apply a variety of z-transforms because the z-transforms are also classes. In fact, the z-transforms are also modeled using

[1] LaSRS++ also contains filters that use fixed-step integrators, include rate and position limiting, etc.

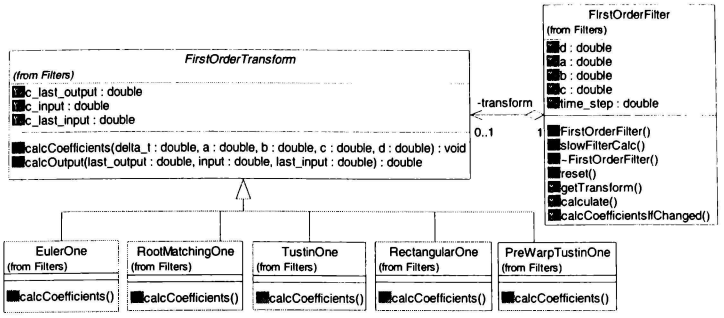


Figure 2 Z-Transform Class Diagram

the strategy design pattern. The filter object is paired with a z-transform when constructed; developers can mix and match filter objects with z-transforms#. These simple z-transforms are algorithms of the form:

Equation 1 Simple Z-Transform

$$y = ax + \sum_{n=1}^{O} \left(b_n x_{t - n\Delta t} + c_n y_{t - n\Delta t} \right)$$

where y is the output, x is the input, O is the order of the transfer function, and a, b, and c are coefficients. The coefficients are derived from the Laplace-space coefficients, the time step, and the z-transform. Since order determines the number of coefficients required, the z-transforms, like the filters, are categorized by order. Thus, a filter object can only be paired with a z-transform of the same order. Figure two shows the first order z-transforms**. The base class, FirstOrderTransform, encapsulates Equation 1 to first order. The base class holds three coefficients, one each for the input, last input (t-Δt), and last output (t-Δt). calcOutput() implements the equation. calcCoefficients() is a polymorphic method; its arguments demonstrate that the coefficients are purely a function of the Laplace-space coefficients and the time step. The sole purpose of the derived class is to define this coefficient function based on a given z-transform.

The Filter hierarchy stores the past values needed in Equation 1. The Filter class stores the last output and last input; all filters, regardless of order, require them. The SecondOrderFilter stores the additional past states that it needs (i.e. input and output at t-2Δt).

Encapsulation

Filter objects are defined when they are constructed. For example,

```
FirstOrderFilter filter_1(timer,
  mode, a, b, c, d, TUSTIN);
SecondOrderFilter filter_2(timer,
  mode, a, b, c, d, e, f, EULER);
```

The arguments are defined as follows: 'timer' is a reference to a class that contains the time step. 'mode' is a reference to an enumeration that represents the current simulation mode (i.e. RESET, HOLD, OPERATE, etc.). The letters a through f are the Laplace domain coefficients. The final argument is an enumeration that specifies the z-transform used by the filter. The argument list does not contain any "scratch" array. The

LaSRS++ filters that utilize fixed-step integrators as their digital algorithm follow the same design. The integrators are encapsulated in classes that can be matched with a filter class.

** The design for the second order transform is identical except that the base class, SecondOrderTransform, contains five coefficients and its methods require two more double arguments.

"scratch" variables are coded into the filter and z-transform classes. A filter object contains the exact number of past values it requires. The z-transform contains the exact number of derived coefficients that it requires. To create the filter, the developer does not need to know much workspace a filter needs nor declare the workspace. The object-oriented design isolates the developer from that level of detail. The workspace variables are "encapsulated" in the class definitions. The object-oriented design eliminates the two potential defects associated with the scratch array in the procedural design: reading and writing past array bounds or more than one filter accessing the same scratch array. Also, the object-oriented design allocates the exact amount of workspace required for a given mix of filters. To avoid the array bounds defect, the procedural design encourages developers to allocate more workspace than necessary for a given mix of filters.

Polymorphism

In object oriented systems, a derived class can be assigned to and accessed through a reference[††] to its base class. Since FirstOrderFilter and SecondOrderFilter inherit from Filter. FirstOrderFilter and SecondOrderFilter objects can be assigned to Filter pointers. For example,

```
Filter* filter_1 =
      new FirstOrderFilter(timer,
      mode, a, b, c, d, TUSTIN);
Filter* filter_2 =
      new SecondOrderFilter(timer,
      mode, a, b, c, d, e, f, EULER);
```

Client code can only invoke methods in the Filter class interface through these pointers. However, polymorphism allows the derived class to redefine the behavior of methods in the base class interface. The reset() and calculate() methods in Filter are polymorphic. Executing filter_1->reset() actually calls FirstOrderFilter::reset(). Likewise, filter_2->reset() actually calls SecondOrderFilter::reset(). In the Filter design, the client does not call the polymorphic functions directly. Instead, the client calls fastFilterCalc(). fastFilterCalc() calls reset() or calculate() as appropriate for the current simulation mode[‡‡]. Thus, the control system code executes the filters as follows:

```
double output_1 =
  filter_1->fastFilterCalc(input_1);
double output_2 =
  filter_2->fastFilterCalc(input_2);
```

The syntax for both filters is identical even though the filters are defined with different transfer functions and different z-transforms. The first line will execute a first order filter using a Tustin z-transform. The second line will execute a second order filter using an Euler z-transform. No part of the filter definition appears in the two statements. Instead, filters are defined when constructed. The developer can change the order of the transfer function, the Laplace-space coefficients, or the digital algorithm without modifying the operational code. The execution of the filter is separate from its definition.

All filter definitions are collected into one block of code within the initialization unit. Viewing all of the definitions is simple. The developer does not have to travel from one filter execution to the next to observe all the definitions. The collected definitions can be viewed as a legend to the filters in the control system. The fact that filters must be named makes this notion more useful. Filter object names can correspond to names on a block diagram facilitating verification. In the procedural design, naming the output of the filter after the name on the block diagram provides a similar link; the block diagram name and the filter definition appear on the same line. However, the developer must still traverse hundreds of lines of code to verify all of the filters.

Being able to cluster the filter definitions also facilitates the creation and maintenance of multiple configurations. To define and manage multiple configurations, only one conditional construct is required in the initialization code. All the configurations and their associated filter definitions appear in the same block of code for easy identification. To add a new configuration, the developer simply extends the conditional construct and populates the new condition with filter construction

[††] A pointer or reference in C++.

[‡‡] In HOLD mode, fastFilterCalc() does not call either method; it returns the last output.

statements. No conditional statements need appear in the operational code; the complexity of the operational code is not increased by multiple filter configurations. The object-oriented design successfully allows the whole filter definition to be configured during initialization. The operational code is no longer forced to redundantly evaluate one-time initialization logic.

The following examples summarize the steps involved in modifying filters and creating a new configuration using the object-oriented design; they represent worst case conditions: order, coefficients, and digital algorithm are changed. The filter order is being increased in the example.

Changing an existing filter

1) Search the initialization code for the filter using its name.
2) Change the filter construction line to construct a filter according to the new definition.

Creating a new configuration

1) Search the initialization code for the conditional construct.
2) Expand the conditional construct for the new configuration.
3) For each filter, construct a filter object of the appropriate type and assign it to its corresponding Filter reference.

The steps for the object-oriented design are fewer than the steps for the procedural design. Changing an existing filter takes two steps in the object-oriented design versus seven in the procedural design. Creating a new configuration requires two one-time steps and one additional step for each filter in the object-oriented design. The procedural design can require eight steps for each filter. The possibility that a step will be neglected or performed incorrectly is reduced in the object-oriented design. Moreover, the steps only affect one code unit, initialization. Thus, there are fewer code units to review, test, and verify. Changes to fewer code units potentially leads to fewer re-compilations. Isolating the changes to the initialization unit also facilitates configuration management of the changes. Compared to the operational code, the initialization unit is usually smaller and changes less frequently because it focuses on one task, defining the variables in the control system. This simplifies the reconciling of changes from multiple developers. All these factors indicate that the object-oriented design is less costly to maintain than the procedural design.

Advantages of Z-Transforms as Objects

The discussion thus far concerns the maintenance of filters whose definition is immutable during operation. Many of the advantages that the object-oriented design holds over the procedural design disappear when the filter is mutable, i.e. the coefficients of the filter change during operation. In this situation, the order and coefficients of the filter must be known at the point of execution. The filter design reflects this restriction. Mutable filter calculations are handled by the slowFilterCalc() method. This method is not polymorphic. It is declared and defined in each derived class; it does not appear in the Filter class interface. The method can only be invoked by accessing it through the derived class. For example,

Initialization Code:

```
FirstOrderFilter filter_1(
  timer, mode, a, b, c, d, TUSTIN);
SecondOrderFilter filter_2(timer,
  mode, a, b, c, d, e, f, EULER);
```

Execution Code:

```
filter_1.slowFilterCalc(
  input, a, b, c, d);
filter_2.slowFilterCalc(
 input, a, b, c, d, e, f);
```

filter_1 and filter_2 are objects of the FirstOrderFilter and SecondOrderFilter classes; they are not references to the base Filter class as in the previous example. They implicitly identify the order of the transfer function wherever they are used. The coefficients appears as arguments to the slowFilterCalc() call; thus, they are also defined wherever slowFilterCalc() is execute.

However, the digital algorithm does not appear at the operational code as it does in the procedural design. The z-transform algorithm that the filter uses can be modified without changing the operational code. Configurations that differ only by the z-transform can still be defined in the initialization code. The procedural design still changes the operational code when the z-

transform is changed. Thus, the object-oriented design can still configure part of the mutable filter (i.e. the z-transform) in the initialization code; the procedural design forces all configuration logic into the operational code.

Polymorphism and encapsulation make this possible. How the z-transform and filter classes interact is a microcosm of how the filter and control system classes interact. The FirstOrderFilter accesses the derived z-transform class only through a reference to the FirstOrderTransform base class. The z-transform is defined only in the FirstOrderFilter constructor. The FirstOrderFilter constructs a z-transform according to the transform enumeration and assigns the z-transform object to a FirstOrderTransform reference. The filter's operational code only uses the reference; thus, the definition of the z-transform does not appear in the filter's operational code. The z-transform is configured once, in the filter's constructor call.

Performance

In a simulation environment, performance is an important factor in analyzing a design. Even though the object-oriented design facilitates maintenance, it must perform adequately to be a good design for simulation. The performance of the LaSRS++ filters was compared against those in LaSRS. The comparison was performed on an SGI Origin 2000. The SGI MIPSPro compiler version 7.2.1.3m was used to compile both the C++ and FORTRAN source. All files were compiled using the '-O2' optimization flag; no other optimization options were included. Performance was analyzed using the SGI Performance Analyzer. The performance analyzer was set to use program counter sampling and the R10000 hardware cycle counter. The test program ran each filter one million times using a predefined set of numbers for input. The performance tests were run ten times to verify consistency of the results. The median cycle count from the ten runs was used to calculate the average cycle count per call; the average was rounded to the nearest integer. The results appear in Table 1.

Table 1 C++ vs. FORTRAN Performance[§§]

Filter Type	Average Cycle Count	
	FORTRAN	C++
First Order Tustin	46	34
Second Order Tustin	50	43

The numbers demonstrate that the object-oriented design has performance comparable to the procedural design[¶¶]. Performance is not sacrificed by using the object-oriented design.

Conclusions

The object-oriented design facilitates maintenance of filters by moving the definition of a filter from its point of execution to the initialization code. In the initialization code, all filter definitions can be collected for easy inspection and modification. Multiple configurations can be created and managed by placing these filter collections into one conditional construct. This "one-time" configuration logic is also efficiently placed in the initialization code. The procedural design can force configuration logic into the operational code where it is redundantly evaluated each frame. The object-oriented design also shields the developer from the need to create and manage workspace data for the filter. Errors associated with active management of the workspace are eliminated. In comparison to the procedural design, the object-oriented design reduces the steps to change an existing filter or add a new configuration. Changes can be made rapidly, with few errors. The benefits of the object-oriented design are obtained without sacrificing performance.

[§§] For both the FORTRAN and C++ source, functions optimized for immutable filters were used, a.k.a. the "fast" filter calculations.

[¶¶] The performance tests were not exhaustive. Other compilers or other optimization flags could change the results. However, the results are sufficient to demonstrate comparable performance.

Bibliography

[1] Booch, Grady. *Object-Oriented Analysis and Design With Applications*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994. ISBN 0-8053-5340-2.

[2] Stroustrop, Bjarne. *The C++ Programming Language*. Addison-Wesley Publishing Company, Reading, MA, 1997. ISBN 0-201-88954-4.

[3] *Working Paper for Draft Proposed International Standard for Information Systems – Programming Language C++*. ANSI X3J16/96-0225, 1996.

[4] Quatrani, Terry. *Visual Modeling With Rational Rose and UML*. Addison-Wesley. Reading, MA, 1998. ISBN 0-201-31016-3.

[5] Stevens, Brian L.; Lewis, Frank L. *Aircraft Control and Simulation*. Addison-Wesley Publishing Company, New York, NY, 1992. ISBN 0-471-61397-5.

[6] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. *Design Patterns: Elements of Object-Oriented Software*. Addison Wesley Publishing Company, Reading, MA, 1995. ISBN 0-201-63361-2.

[7] Leslie, R.; Geyer, D.; Cunningham, K.; Madden, M.; Kenney, P.; Glaab, P. *LaSRS++: An Object-Oriented Framework for Real-Time Simulation of Aircraft*. AIAA-98-4529, Modeling and Simulation Technology Conference, Boston, MA, August 1998.

CHALLENGES IN INTEGRATING COMMERCIAL OFF-THE-SHELF (COTS) SOFTWARE INTO A REAL-TIME SIMULATION ENVIRONMENT

Joseph Nalepka, Daniel Caudill, Larry Mutschler
Air Force Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433

Christopher Buell, Lucy Garcia, David O'Quinn
Veridian
Dayton, Ohio 45431

Abstract

Simulation Based Acquisition (SBA) is becoming a more prevalent method for planning, designing, integrating, and testing advanced weapon systems or other products. The end result of this process, in theory, is a product that is produced faster, cheaper and more reliably than its predecessors. It is critical that the environment being simulated in the SBA process accurately model the conditions under which the SBA product will be operating. Consequently, program managers must make a decision as to what types of components are necessary for the simulation as well as how these types of components will be developed. Commercial Off-the-Shelf (COTS) software products are being used to fill those simulation needs, maintain simulation fidelity, and to help meet schedule and cost constraints. An ongoing simulation program being conducted at the Air Force Research Laboratory (AFRL) simulation facility located at Wright-Patterson Air Force Base successfully integrated COTS software products into a real-time, simulation architecture. During the simulation development process, several hardware and software challenges were encountered that made integrating the COTS software products within the simulation architecture very difficult. Once these challenges were overcome, the end result was a very realistic and cost effective simulation that will be used to assess the feasibility of technologies being slated for a future tactical aircraft.

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

Introduction

An initiative that is becoming more prevalent within the DoD community is Simulation Based Acquisition (SBA). The goal of this process is to plan, design, and test a weapon system or other product entirely on a computer. The end result is a product that is produced faster, cheaper and more reliably than its predecessors. In addition, because simulation is used early and often during this process, personnel will be receiving the necessary training on how to operate and maintain the product when it is fielded[1].

Because simulation is the critical component of the SBA process, it is necessary for the simulation to accurately represent the operating environment of the system that is being developed and evaluated. Take, for example, a proposed aircraft system that will be used as a strike vehicle in a hostile environment. To test and evaluate this system, components such as the radar, on-board weapons and the digital threat environment need to be modeled in accurate detail. If the level of simulation fidelity is not sufficient, errors could be made in the selection of components to be implemented on that future weapon system.

In the past, the inclination was for simulation engineers to develop the necessary simulation components from scratch. There are several drawbacks to this approach. First, in-house expertise may not exist or be very limited for developing the software. The end result is a component that may be less capable than expected. Second, even if the in-house expertise does exist, it may take many man-months or even man-years to generate a functionally acceptable software module. If several of these modules must be created, both the cost and the time it takes to develop the simulation increase. As a result,

simulation engineers and program managers are faced with a very tough question: How does one develop an SBA simulation that satisfies a program's functional and schedule requirements while keeping development costs down?

One approach that is gaining greater acceptance is to use Commercial Off-the-Shelf (COTS) software products to fill these simulation needs. COTS software is any pre-built software component that is used as a part of the system under development. These components could include complete applications or services, subsystems, libraries of subroutines, abstract data types (ADT's), or classes[2]. By using an off-the-shelf software tool or package, the cost and time for simulation development can be greatly reduced. However, the use of a COTS product is not always as simple as "plug and play." There are challenges that can occur when integrating one or more COTS products into a real-time simulation environment. The focus of this paper will be to identify both the benefits and drawbacks of using COTS software. In addition, an SBA example will be presented to illustrate some of the hardware and software challenges that one may encounter when integrating COTS software into a simulation architecture.

Benefits and Drawbacks of COTS Software

One of the major benefits of COTS software is that it can significantly reduce the initial simulation development cost. Although the purchase price of COTS software packages can be high, there is a significant long-term cost saving in comparison to developing a similar product in-house. There is, however, a drawback to purchasing COTS software. This is the recurring cost associated with this software. An annual support agreement such as the Software Update Service (SUS) is usually required between the customer and the vendor for obtaining software support. In addition, upgrades or enhancements that are not covered under a SUS come at an additional cost to the customer.

Another benefit of COTS software is that the vendor is always adding extra capability or making performance improvements to keep up with the functional demands of its customers. Also, because the product is being used by several different customers with different applications, much of the COTS software functionality is continuously being validated by the customers, with software "bugs" being reported back to the vendor for correction. The end result is a product that is typically better designed, more capable, and more reliable than a product that is custom-built[1].

There are some additional benefits to using COTS software. First, the corporate or in-house knowledge emphasis can be on how the software works rather than on how it was put together. This can be done either through sharing in-house expertise or by more formal means such as training seminars that are provided by the vendor. Second, if a large customer base exists for the software, experiences with using the software can be shared through "User's Group" Meetings to not only better understand the COTS application but also to influence the type of enhancements that are to be added to future software releases. Third, if there is a need for a specific software enhancement, the vendor can usually provide on-site support to implement this new software feature. This, however, will usually come as an additional cost to the customer. Finally, with the proper training and experience with the software, its use and thus its cost can be spread across many other different simulation programs.

The biggest drawback to using COTS software is integrating it into the virtual simulation architecture. During integration, one must find a way for the COTS software to communicate with other COTS products and the in-house developed software components that will comprise the overall simulation environment. This is usually done through a custom designed interface or "wrapper." A wrapper is a shell or buffer between the COTS software application and the simulation. Its purpose is to not only transfer data from the application to the simulation and back but also ensure to that the values being transferred are within certain bounds to avoid execution errors[6]. Because source code or other technical information about the COTS software's design is not usually available to the customer, it is sometimes very difficult to design an efficient wrapper. Because of this "black box" COTS software design, the customer must sort through on-line documentation or user's manuals to get an understanding of the input and output requirements of the COTS software. In situations where a simulation is using several COTS software packages, the design and integration of these wrappers can be difficult and time consuming.

Another drawback of using COTS software involves the level of fidelity of the software package. Because COTS software is designed to reach a very broad customer base, the software may have numerous features that are modeled at a moderate level of detail rather than only a few features modeled at a very high level of detail. Because the COTS software is designed as a black box, the customer does not have access to the source code and is unable to add enhancements to improve the level of fidelity[6].

In addition to a fidelity limitation, one may also be faced with a capability limitation. In some instances, when an upgrade is made to a COTS software package, a capability that may have existed in the previous version of the software and that is required for the simulation may no longer exist for the new version[4]. Consequently, one may be forced to forgo the upgrade in order to maintain the current capability. However, the vendor may no longer support the older version of the COTS software and thus technical support for the older software versions may no longer exist[5].

As a result of the fidelity and capability limitations, the program manager is faced with a very important trade-off issue: Does one use a less capable product or does one develop an in-house product at the risk of increased program cost and schedule delays to obtain the desired functionality? It is at this point that the program manager must further investigate the requirements of the simulation in order to determine if the lack of fidelity or capability will affect the results of the study before making a decision on the use of a particular COTS software product.

A final drawback of COTS software is the execution platform for the software. In some cases, the COTS software is computationally intensive and may require a computer with fast processing power. Also, the amount of memory (RAM) that may be required for a COTS software module may be quite large. This is due in large part to the complexity of the software and its methods for allocating memory during execution. As a result, additional memory may be required for the computer. Finally, these software packages may require more than one processor to execute. Consequently, a computer with many CPU's may be required to handle the entire simulation environment. The end result is that additional cost is incurred, on top of the software purchase, to upgrade or purchase a computer that can handle the computational requirements of the integrated simulation environment.

Simulation Example

The SBA process is currently being used within the Air Vehicles Directorate of the Air Force Research Laboratory (AFRL) located at Wright-Patterson Air Force Base to support the design and development of a future tactical aircraft. This simulation effort has three objectives. First, to provide the tactical aircraft's program office with an "expert" opinion regarding the utility of proposed technologies to be incorporated into the vehicle as well as the value of these technologies to the pilot. Second, to provide the vehicle advisory panel with insight that will enable them to make informed decisions regarding key operational requirements for the vehicle. Third, to use the advisory panel's requirements to refine the vehicle requirements that will be included in the overall requirements document for the vehicle. This final document is what the airframe manufacturer will use when designing the vehicle. The end result of this process will be a vehicle that has the "correct" technologies the first time around, thus avoiding costly design changes or system changes once the vehicle is fielded.

There are two types of missions currently under investigation for this SBA effort. These are the Unit Interdiction (UI) and Choke Point Interdiction (CPI) missions. The UI mission objective is to strike a mechanized brigade in mountainous terrain. The CPI mission is broken down into two parts. The first is a Suppression of Enemy Air Defenses (SEAD) mission in which the objective is to attack a fixed Surface-to-Air Missile (SAM) site. The second is a Time Critical Target (TCT) mission in which the objective is to "seek and destroy" a mobile target.

The types of missions that are being assessed and the vehicle subsystems that are being evaluated require that the simulation have accurate representations of the threat battle space, radar and imagery, aircraft dynamics, and cockpit instrument displays. In addition to the strict fidelity requirements, the program was under a very tight development schedule. Simulation development began in February of 1998 and had to be completed for the first set of piloted testing by August of 1998. At the start of the development process, a number of the software components

needed to meet the simulation's requirements that were available in-house were not of high enough fidelity or lacked certain critical functionality. Because the amount of time between development and testing was very short, it was determined that modifying the existing software or creating new software modules to meet the requirements was not possible. Consequently, it was necessary to purchase several COTS software products to provide the necessary simulation capability. The primary COTS products that were utilized are shown in Table 1.

Product	Source	Purpose
Camber Radar Toolkit	Camber Corporation	Provide real-beam map and SAR imagery
Large Area Database Manager (LADBM)	Paradigm Simulation Inc.	Provide visual display management
SensorVision	Paradigm Simulation Inc.	Provide infrared (IR) imaging
Simulated Warfare Environment Generator (SWEG)	U.S. Navy	Provide Digital Threat Environment

Table 1 – COTS Software Used in the AFRL Simulation

The COTS software products that were chosen were based on research done by the program management team prior to simulation development. Since Paradigm Simulation Inc. was under contract to develop the terrain database for this program, it was a natural extension to use their products for the image-related functions. The Camber Radar Toolkit was chosen because of its proven ability to accurately depict real-beam radar operations and Synthetic Aperture Radar (SAR) image representations. Finally, the Simulated Warfare Environment Generator (SWEG) was chosen because of its ability to model a mission level simulation environment consisting of threat tactics, communications, and operations at the level of fidelity necessary for accurately assessing the operational effectiveness of the tactical vehicle in a realistic battle space environment.

Another set of requirements imposed on the simulation was that it must support two independently operated vehicles. This requirement, along with the computational requirements of the simulation software, drove the selection of the computational hardware. Two, fourteen-processor Silicon Graphics ONYX-2 computers (one for each simulation vehicle) were utilized. Because both vehicles would be working a coordinated mission, a shared memory interface (SCRAMNet) was required and implemented between both of the machines. Information between the two machines was also transferred along the SCRAMNet interface for use with the Virtual Battlefield Management System (VBMS) located in the simulation control room. The computer architecture for one of the simulation vehicles is shown in Figure 1.

During the six-month simulation development effort, the simulation team was faced with many hardware and software challenges when integrating the COTS software into the real-time simulation environment. The remaining focus of this paper is to discuss those challenges and the steps that taken to resolve them.

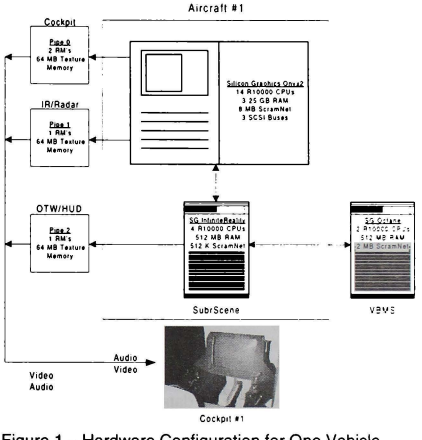


Figure 1 – Hardware Configuration for One Vehicle

Software Challenges

During the simulation development process, there were several software challenges that were encountered and resolved by the simulation team.

Interface Design

Designing and implementing a "wrapper" for the COTS software is a very formidable challenge. As mentioned previously, a wrapper is an interface mechanism between the COTS software and the rest of the simulation environment. This wrapper must supply the necessary input parameters to the COTS software and retrieve from the COTS software the parameters that are needed by the

rest of the simulation. There are several details in this process that must be considered.

The first detail is to determine if the inputs required by the COTS software are available from the simulation. In many cases, these inputs are usually constants that define certain characteristics or operating modes of the COTS software. These parameters can sometimes be defined directly in the wrapper. However, information such as an aircraft's state or a cockpit switch setting must come from other modules within the simulation environment and usually through a shared memory partition. It is the job of the wrapper to access this shared memory and extract the required input parameters. If this information is not available, the simulation software must be modified in such a manner that the required input data is computed and made available to the wrapper.

A second detail that must be considered is to make sure that the information being passed to the COTS software is of the proper format. Within the wrapper are a series of "converters" that assure that the correct units are assigned to the values passed into the COTS software. In addition, these converters are also responsible for making sure that the input parameters are in the correct coordinate frame being used by the COTS software. For this simulation, both SWEG and Sensor Vision expected not only orientation information but also position information to be in a coordinate frame that was different from the rest of the simulation.

The final detail that must be considered when designing a wrapper is data management. The end result of executing the COTS software module is the data that is generated by this module. It is the job of the wrapper to assure that this data is passed on to the rest of the simulation through shared memory or other mechanisms. Because modules within a simulation environment can be of the C, C++, ADA, or FOTRAN programming languages, passing data to these modules becomes difficult. For example, a Boolean value in FORTRAN is usually represented as either TRUE of FALSE whereas in C it is represented as either zero or non-zero. In addition, it is not possible to call ADA modules from C or FORTRAN modules and thus using a shared "common block" is not an option. As a result, an interface mechanism must be written in conjunction with the wrapper that will place data in a shared memory block that can be accessed by all software modules regardless of language.

Software Validation

Once the COTS software has been integrated into the simulation environment, the next challenge is to verify that it is functioning correctly. For in-house developed software modules, there usually exists a set of "check cases" or baseline results that can be used to verify if a module was properly integrated into the simulation environment. For COTS software, this is usually not the case. Proper operation is usually verified using a demonstration program that is provided by the vendor. Although this is a good mechanism for determining if the software was installed correctly, it will not work for determining its effectiveness in the simulation environment. This is true for two reasons. First, this demonstration program usually operates in a stand-alone mode so one is unable to interface it with the simulation environment. Second, because it is not interfaced with the simulation environment, one is unable to test out the functionality of the wrapper. Consequently, the question to ask is how does one validate the operation of a COTS software package?

The best way to answer this question is with an example. One of the more challenging COTS software packages to validate for this simulation program was the Camber Radar Toolkit. This package was difficult to validate because not only did it require numerous parameters to define the characteristics of the radar but it was also required to operate in two different modes: a Real Beam Ground Map (RBGM) mode and a SAR mode. As a result, the validation of the CAMBER radar model meant having to separately validate the two different modes of operation.

The purpose of the RBGM was to give the pilot a representation of terrain features such as rivers, mountains and roads. Since maps of the visual database were available, the RBGM operation was verified by pointing it at a certain area within the database and comparing the image against the features on the database maps. The same comparison was done with the SAR mode. However, with this mode, one is able to discern additional features such as buildings or vehicles. So, in addition to verifying the terrain characteristics, objects were placed and removed from the scene manually and checked with the SAR mode image to see if they appeared in the

correct place. This movement of both the area of interest and the objects not only verified the parameters of the radar, but, because the simulation was issuing the movement commands, it helped validate the operation of the wrapper.

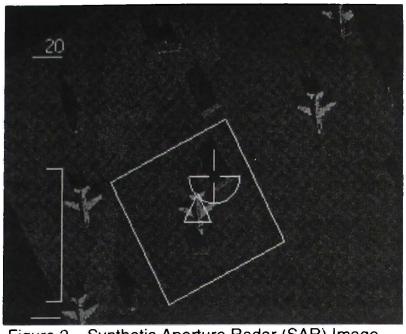


Figure 2 – Synthetic Aperture Radar (SAR) Image

In addition to validating the operating modes of the Camber Radar Toolkit, there was also a need to validate the physical appearance of the images in the SAR display. In a SAR display, objects that are very bright are ones that reflect light very well such as tanks, aircraft or other metallic objects. Objects that are less bright are ones that do not reflect light very well such as trees, buildings or other non-metallic objects. An example of this can be seen in Figure 2. The amount of brightness that an object exhibits is based on the "material properties" that are defined for that object in the radar model. Although the distinction between which objects will be brighter than others is easy to make, the amount of adjustment that needs to be made to the material properties in order for the object to have a "real world" appearance is difficult. As a result, it was necessary to consult with several pilots who were experienced with SAR imagery to help with the adjustment of the material properties. Although this was a purely subjective method for validating the appearance of objects within the SAR display, it was an effective method and the display itself was well received by the evaluation pilots during testing.

This is just one example of how to validate a COTS software package. In general, the type of validation method that is used will be based on the type of COTS software that is being utilized. However, a few validation rules of thumb can be followed that can apply to any COTS software package. First, one should pick a scenario or geographic area that is the same throughout the validation process. By doing this, one will be familiar with the way things should appear or function and any deviation from this would be an indication that something is not functioning correctly. Second, one should validate each COTS software package independently and once it is validated, the simulation should be baselined. This will provide a check case in itself when new simulation modules are added and validated within the simulation architecture. Finally, a test matrix should be created that contains not only the optimal operating conditions but also the worst case conditions for the COTS software package. This will not only verify the complete operation of the software but also test its run-time performance under the most computationally intensive conditions.

Task Management

Once the COTS software was integrated into the simulation environment, the next challenge was task management. Task management refers to the "scheduling" or distribution of the simulation modules or processes amongst the various processors on the simulation computers. Each simulation task or process, whether it be a COTS module or an in-house developed module, requires a certain amount of execution time on a specific computer processor unit (CPU). Consequently, one must decide how to distribute all of the simulation tasks across the CPU's while maintaining real-time performance.

The first step in this process is to identify the software modules that have the largest CPU requirement. Because of their complexity and detail, these tend to be the COTS software modules. For example, the Camber Radar Toolkit requires up to eight dedicated CPU's to maintain optimal, run-time performance. Of course, if one allows the COTS software to utilize it's "optimal" number of CPU's, there would be very few CPU's left for the rest of the simulation software. The issue now becomes a tradeoff between execution speed and resource management.

In general, many of these COTS software packages will not be running at their maximum capability and thus will not require all of the recommended CPU power. As a result, one can

reduce the number of dedicated CPU's for each package. However, execution speed may decrease because the parallel processing capability of the COTS software is reduced. Unfortunately, there is no rule of thumb for determining how many CPU's are actually needed to do the job. This determination must be done through experimentation to see where the least amount of delay is experienced for a specific CPU allocation. Although the Camber Radar Toolkit required up to 8 CPU's for maximum efficiency, it was discovered that only 4 CPU's were actually required to maintain a high level of execution performance for this simulation.

The next step in the task management process is to determine how to allocate the remaining simulation tasks amongst the remaining CPU's. This can be done by first determining which simulation tasks must be run synchronously with the simulation and which ones can be run asynchronously. A synchronous task is one that requires a deterministic, real-time update rate and typically includes such things as high-fidelity vehicles, out-the-window displays, cockpit interpreters, and data recording devices. An asynchronous task is one that does not require a deterministic, real-time update rate and is basically allowed to run as fast as the CPU will allow. These are usually tasks that do not change rapidly during a simulation or are not time critical and typically include such things as head-down displays, cockpit sounds, or multi-function displays. Since asynchronous tasks are not time critical, they can share a CPU with other asynchronous tasks on a time available basis.

Once the synchronous task requirements are identified, the final step in the task management process is to determine the amount of processing time required for each synchronous task. Because the number of CPU's available for the synchronous tasks are limited, several synchronous tasks would have to execute on the same CPU. By knowing the processing time of each task, one can distribute the tasks across the processors in such a way that a real-time update rate is maintained and no frame overruns occur. It is assumed, of course, that the ordering of the tasks on a specific CPU is done in such a way that software dependencies are taken into account. For example, a simulation task that requires the current state of a vehicle must follow the simulation tasks that compute these quantities.

As was the case with software validation, there is not a set procedure for determining how to allocate tasks across a CPU. Much of this is done through experimentation and analyzing the delays associated with a specific configuration. If the appropriate research was done prior to the purchase of both the computer hardware and the COTS software, finding a configuration that will provide reasonable execution performance should not be a problem. If this is not the case, the result will be to either purchase more processing power for the computers or eliminate software modules in order to reduce the processing requirements.

Hardware Challenges

In addition to the software challenges encountered by the simulation team, the team also encountered two major hardware challenges when integrating COTS software products into the simulation architecture.

Shared Memory

The first hardware challenge was the selection of shared memory for the simulation computers. For COTS software, the shared memory requirements are usually defined on a product specification sheet. Because the shared memory requirements of each COTS software product are known, one approach for selecting the amount of shared memory for the simulation computers is to add together the COTS software shared memory requirements and the memory requirements of the other simulation software modules. Although this approach is the simplest and the most obvious, it is not the most complete.

Although COTS software products tend to use the most shared memory and thus drive the overall memory requirements of the computer system, other factors must also be taken into account. The first is the memory requirement of the computer's operating system (OS). One must be aware that the OS does require a certain amount of shared memory in order for it do such things as disk management and file manipulation. Depending on the configuration of the computer system, additional OS applications may also be active and will also require shared memory resources.

A second factor that must be taken into account is the workload of the computer above and beyond what is designated for the simulation. This extra workload consists of other applications executing on the simulation computers such as software

editing or software compilation. All of these additional "background" tasks require shared memory for their execution and thus contribute to the overall memory requirements of the computer system.

Of all the possible hardware challenges that one could encounter during simulation development, this is probably the easiest one to overcome. The solution to this challenge is to just purchase more shared memory for the computer. Although this is the easiest approach, it does come with two big risks. The first risk is cost. Because a simulation project and its related equipment are confined to a very strict operating budget, extra costs incurred due to planning oversight may be very difficult to justify and take a very long time to resolve. The end result is a program that is over budget and possibly delayed until the additional shared memory is purchased for the computer.

The second and potentially most damaging risk of underestimating the shared memory requirements of the computer is the selection of an undersized machine. The amount of shared memory that can be added to a computer is strictly computer architecture dependent and can range anywhere from a few megabytes to several gigabytes. If the shared memory requirements of a computer system are grossly underestimated, one may find that there is not enough memory expansion available to accommodate the overall simulation requirements. As a result, one may be faced with several undesirable solutions such as the purchase of a new computer system that will handle the shared memory requirements of the simulation, the distribution of the simulation software tasks across several different computers to reduce the memory requirements on any one machine, or simply "doing without" a few or several simulation applications. The end result can be a more complicated, costly and potentially less capable simulation than originally planned.

Processing Performance

The second hardware challenge that was encountered involved finding a way to speed up the execution rate of a COTS software product; specifically the Large Area Database Manager (LADBM) utility. The purpose of this COTS software product was to manage the database of the visual display that was used for the simulated infrared imagery. This display management process involved the processing of only those sections of the visual database that corresponded to the area that was being surveyed by the pilot. Because of the size of the visual database, it could not be read and stored into memory. Instead, the pieces of the database that were needed for the display were read into the LADBM from the computer's hard drive. This is a very slow and inefficient process and will produce significant delays in the rendering of the display image.

It was not possible to improve the speed performance of the LADBM by modifying the COTS software itself. Instead, the disk input/output performance was addressed by taking advantage of a Silicon Graphics Computer hardware feature called "drive striping." Drive striping is a process in which two or more hard disk drives are configured such that the computer hardware perceives them as a single drive. These drives can be on the same or different SCSI busses and can be of different sizes. For this simulation, two, nine-Gigabyte hard disks were striped together and placed on the same SCSI bus. During the striping process, the visual database used by the LADBM was divided between the two hard disks. In order to maintain the data format in the database file, each block in the original database file was alternately written to each of the hard disks. In other words, block one of the file was written to disk one, block two was written to disk two, block three was written to disk one, and so on until the entire database file was replicated across the two disk drives. Once this process was complete, the computer perceived only one, Eighteen-Gigabyte hard drive containing the entire database on a single SCSI bus.

The question one may ask is how does this striping process help improve the speed performance of the LADBM? Because the visual database is written in alternating blocks and the LADBM knows how many blocks of data it needs for rendering the correct image, the computer can read in the data from both hard disks at the same time rather than from just one. For example, if the image to be rendered requires the first two blocks of the visual database, the original configuration would require that the blocks of data be read in one at a time. Under the striped configuration, the computer will access both hard disks at the same time and read in both blocks of data simultaneously and merge them together at the end of the process. This effectively improves the disk access performance by a factor of two. In

addition, because both disks are on the same SCSI bus and only the LADBM will be accessing data on this bus, there will not be a SCSI device access delay caused as a result of another application utilizing this SCSI device. As a result of this hardware architecture modification, the delays in disk access and thus image rendering in the cockpit were greatly reduced.

Conclusions

The use of COTS software in a simulation environment can be very valuable when developing a simulation under very strict cost, manning, and schedule constraints. This software can not only provide a simulation with an increased level of fidelity but also, because it is generic in nature, it can be used for other simulation development efforts as well. However, challenges do exist when attempting to integrate this software into the existing simulation hardware and software architecture. These challenges include the design of wrappers to interface the COTS software with the rest of the simulation, verifying that the COTS software is functioning correctly within the simulation environment, the distribution of software tasks across the CPU's of the computer, the selection of memory for the computer, and developing methods for improving processing performance. Once these challenges are overcome, the end product will be a simulation environment that is very flexible and has sufficient fidelity for testing and evaluating present and future technology.

References

1. "Simulation Based Acquisition: What Will Computers Do Next?", *Acquisition Reform Today Newsletter*, November/December 1998, pp. 5.

2. Dean, John C. and Vigder, Mark R., "System Implementation Using Commercial Off-the-Shelf (COTS) Software", National Research Council of Canada, Report Number 40173.

3. Vigder, Mark R., "An Architecture for COTS Based Software Systems", National Research Council of Canada, Report Number 41603.

4. Dean, John C. and Vigder, Mark R., "An Architectural Approach to Building Systems from COTS Software Components", National Research Council of Canada, Report Number 40221.

5. Abts, Chris and Boehm, Barry, "COTS Integration: Plug and Pray?", *IEEE Computer*, January 1999, pp. 135-138.

6. Voas, Jeffrey M., "Certifying Off-the-Shelf Software Components", *IEEE Computer*, June 1998, pp. 53-59.

AUTHOR INDEX

Ahmadi, R., 45-MST-9
Akin, D., 30-MST-6
Allen, S., 45-MST-9
Baker, M., 30-MST-6
Bendall, C., 13-MST-1
Berglund, E., 13-MST-1
Bertin, R., 29-MST-5
Betin, A., 44-MST-8
Bezdek, W., 13-MST-1
Biezad, D., 69-MST-12
Blake, W., 44-MST-8
Bluff, R., 13-MST-1
Brett, B., 14-MST-2
Buell, C., 71-MST-14
Burdun, I., 43-MST-7
Cardullo, F., 70-MST-13
Carignan, C., 30-MST-6
Carrijo, D., 69-MST-12
Catterall, R., 15-MST-3
Caudill, D., 71-MST-14
Choi, K., 44-MST-8
Chung, W., 15-MST-3, 28-MST-4, 70-MST-13
Cunningham, K., 71-MST-14
Davis, P., 44-MST-8
de Bruijn, R., 45-MST-9
Demidov, V., 15-MST-3
Demyanov, M., 69-MST-12
Dillaber, E., 71-MST-14
Dow, D., 14-MST-2
Drews, M., 43-MST-7
Dudek, H.-L., 15-MST-3
Edmiston, W., 45-MST-9
Eguchi, H., 69-MST-12
Ehernberger, L., 43-MST-7
Elghandour, E., 69-MST-12
Fischenberg, D., 69-MST-12
Fitzhugh, G., 15-MST-3
Forman, D., 43-MST-7
Funabiki, K., 13-MST-1
Garcia, L., 71-MST-14
Geyer, D., 71-MST-14
Gindin, Y., 13-MST-1
Gingras, D., 44-MST-8
Glaab, P., 71-MST-14
Go, S., 69-MST-12
Green, D., 13-MST-1
Grunwald, A., 14-MST-2
Hansson, B., 13-MST-1
Hashii, W., 69-MST-12
Haug, E., 70-MST-13
Hellmann, G., 43-MST-7
Hess, R., 15-MST-3, 28-MST-4
Hoagland, D., 14-MST-2
Hosman, R., 14-MST-2
Houck, J., 70-MST-13
Idan, M., 14-MST-2
Innocenti, M., 43-MST-7
Itano, K., 29-MST-5
Johnson, D., 43-MST-7
Jones, S., 69-MST-12
Kenney, P., 71-MST-14
Kim, C.-J., 14-MST-2
Kim, I., 45-MST-9
Kiso, M., 69-MST-12
Kolkailah, F., 69-MST-12
Krivtsov, V., 43-MST-7
Krozel, J., 14-MST-2
Lane, J., 30-MST-6
Lee, S., 45-MST-9
Lehmer, R., 15-MST-3
Leite Filho, W., 44-MST-8, 69-MST-12
Leslie, R., 71-MST-14
Lewis, W., 15-MST-3
Lim, C.-H., 45-MST-9, 69-MST-12
Looye, G., 44-MST-8
Madden, M., 71-MST-14
Mallaco, L., 69-MST-12
Martin, E., 14-MST-2
McKeehen, P., 69-MST-12
Menke, T., 43-MST-7
Mikula, J., 70-MST-13

AUTHOR INDEX

Miles, E., 44-MST-8
Mittlesteadt, D., 43-MST-7
Monterrubio, L., 69-MST-12
Mook, D., 44-MST-8
Munger, A., 69-MST-12
Mutschler, L., 71-MST-14
Myatt, J., 44-MST-8
Nahon, M., 30-MST-6
Nalepka, J., 71-MST-14
O'Grady, T., 13-MST-1
O'Quinn, D., 71-MST-14
Obana, K., 69-MST-12
Otsuka, R., 29-MST-5
Papelis, Y., 45-MST-9
Park, J., 69-MST-12
Park, S.-J., 14-MST-2
Peleg, A., 13-MST-1
Peters, D., 28-MST-4
Pinto, P., 44-MST-8
Pollini, L., 43-MST-7
Povstyanoy, V., 45-MST-9
Prasad, J., 28-MST-4
Ramires, T., 43-MST-7
Reid, L., 14-MST-2
Richart, D., 43-MST-7
Ro, K., 45-MST-9, 69-MST-12
Rodchenko, V., 70-MST-13
Rodgers, S., 43-MST-7
Ronchieri, E., 43-MST-7
Rubin, L., 45-MST-9
Rufov, I., 70-MST-13
Rutten, A., 70-MST-13
Ryshenko, A., 44-MST-8
Sachs, G., 15-MST-3
Sadovnychiy, S., 43-MST-7, 44-MST-8
Sattler, D., 14-MST-2
Schick, F., 13-MST-1
Schleicher, D., 14-MST-2
Schmidt, D., 43-MST-7
Schroeder, J., 28-MST-4
Schroeder, L., 69-MST-12
Sennes, U., 15-MST-3
Sharf, I., 69-MST-12
Shiomi, K., 29-MST-5
Shome, S., 70-MST-13
Smith, P., 43-MST-7
Smith, R., 45-MST-9, 70-MST-13
Solis, D., 45-MST-9, 70-MST-13
Spuhl, K., 43-MST-7
Stadler, B., 45-MST-9
Subr, R., 28-MST-4
Telban, R., 70-MST-13
Tenoort, S., 13-MST-1
Theunissen, E., 45-MST-9
Thomas, T., 29-MST-5
Thompson, N., 30-MST-6
Tigges, M., 30-MST-6
Tran, D., 70-MST-13
Trapp, T., 13-MST-1
Turner, J., 45-MST-9
Vanhoogstrate Sr., J., 45-MST-9
Vaughan, W., 43-MST-7
Wehrle, B., 45-MST-9
White, A., 70-MST-13
Wing, D., 44-MST-8
Xin, H., 28-MST-4
Yakimenko, O., 15-MST-3
Yashin, Y., 70-MST-13
Yavrucuk, I., 28-MST-4
Yoon, S., 14-MST-2
Zaichik, L., 70-MST-13
Zarkesh, A., 45-MST-9
Zeh, J., 43-MST-7
Zeyada, Y., 15-MST-3
Zipfel, P., 13-MST-1



For permission to copy or republish, contact:
American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive
Suite 500
Reston, VA 20191–4344

ISBN
1-56347-301-1